Strategic Algorithms

by

Evdokia Velinova Nikolova

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

Author .... [illegible] ....
Department of Electrical Engineering and Computer Science
September 4, 2009

Certified by .... [illegible] ....
David R. Karger
Professor of Computer Science
Thesis Supervisor

Accepted by .... [illegible] ....
Terry P. Orlando
Chairman, Department Committee on Graduate Students

На мама, с обич.

Abstract

Classical algorithms from theoretical computer science arise time and again in practice. However, practical situations typically do not fit precisely into the traditional theoretical models. Additional necessary components are, for example, uncertainty and economic incentives. Therefore, modern algorithm design is calling for more interdisciplinary approaches, as well as for deeper theoretical understanding, so that the algorithms can apply to more realistic settings and complex systems.

Consider, for instance, the classical shortest path algorithm, which, given a graph with specified edge weights, seeks the path minimizing the total weight from a source to a destination. In practice, the edge weights are often uncertain and it is not even clear what we mean by shortest path anymore: is it the path that minimizes the expected weight? Or its variance, or some another metric? With a risk-averse objective function that takes into account both mean and standard deviation, we run into nonconvex optimization challenges that require new theory beyond classical shortest path algorithm design. Yet another shortest path application, routing of packets in the Internet, needs to further incorporate economic incentives to reflect the various business relationships among the Internet Service Providers that affect the choice of packet routes.

Strategic Algorithms *are algorithms that integrate optimization, uncertainty and economic modeling into algorithm design, with the goal of bringing about new theoretical developments and solving practical applications arising in complex computational-economic systems.*

In short, this thesis contributes new algorithms and their underlying theory at the interface of optimization, uncertainty and economics. Although the interplay of these disciplines is present in various forms in our work, for the sake of presentation we have divided the material into three categories:

1. In Part I we investigate algorithms at the intersection of *Optimization* and *Uncertainty*. The key conceptual contribution in this part is discovering a novel connection between stochastic and nonconvex optimization. Traditional algorithm design has not taken into account the risk inherent in stochastic optimization problems. We consider natural objectives that incorporate risk, which turn out equivalent to certain nonconvex problems from the realm of continuous optimization. As a result, our work advances the state of art in both stochastic and in nonconvex optimization, presenting new complexity results and proposing general-purpose efficient approximation algorithms, some of which have shown promising practical performance and have been implemented in a real traffic prediction and navigation system.

2. Part II proposes new algorithm and mechanism design at the intersection of *Uncertainty* and

Economics. In Part I we postulate that the random variables in our models come from given distributions. However, determining those distributions or their parameters is a challenging and fundamental problem in itself. A tool from Economics that has recently gained momentum for measuring the probability distribution of a random variable is an *information* or *prediction* market. Such markets, most popularly known for predicting the outcomes of political elections or other events of interest, have shown remarkable accuracy in practice, though at the same time have left open the theoretical and strategic analysis of current implementations, as well as the need for new and improved designs which handle more complex outcome spaces (probability distribution functions) as opposed to binary or n-ary valued distributions. The contributions of this part include a unified strategic analysis of different prediction market designs that have been implemented in practice. We also offer new market designs for handling exponentially large outcome spaces stemming from ranking or permutation-type outcomes, together with algorithmic and complexity analysis.

3. In Part III we consider the interplay of optimization and economics in the context of network routing. This part is motivated by the network of autonomous systems in the Internet where each portion of the network is controlled by an Internet service provider, namely by a self-interested economic agent. The business incentives do not exist merely in addition to the computer protocols governing the network. Although they are not currently integrated in those protocols and are decided largely via private contracting and negotiations, these economic considerations are a principal factor that determines how packets are routed. And vice versa, the demand and flow of network traffic fundamentally affect provider contracts and prices.

   The contributions of this part are the design and analysis of economic mechanisms for network routing. The mechanisms are based on first- and second-price auctions (the so-called Vickrey-Clarke-Groves, or VCG mechanisms). We first analyze the equilibria and prices resulting from these mechanisms. We then investigate the compatibility of the better understood VCG-mechanisms with the current inter-domain routing protocols, and we demonstrate the critical importance of correct modeling and how it affects the complexity and algorithms necessary to implement the economic mechanisms.

Acknowledgements

My advisor David Karger has been a constant source of creative ideas and interesting research problems. I thank him for all that he has taught me over the years. I thank the other two members of my thesis committee Michel Goemans and John Tsitsiklis for inspiring discussions and insightful suggestions. I owe special thanks to Dina Katabi, Asu Ozdaglar and Ronitt Rubinfeld for being wonderful role models and for their support and encouragement throughout my graduate student years at MIT.

Much of my research direction and perspective have been shaped and inspired by outstanding researchers who have given me invaluable guidance and encouragement along the way. Since they are too many to list individually, allow me to thank them together: my coauthors and collaborators; the faculty, postdocs and fellow students in the Theory of Computation group at MIT; the faculty and researchers who invited me to their institutions and gave me the exciting opportunity to present my work; and my colleagues with who we have shared animated research discussions. I have learnt a lot from everyone's experience and perspective.

In a discussion of my work with Ashish Goel, he coined the term *strategic algorithms*, which later became the title of this dissertation.

I thank Be, Joanne, Kathy, Marty and Patrice for their help and support, as well as the MIT EECS Graduate Office including Peggy Carney, Marilyn Pierce and especially Janet Fischer for going out of her way to answer my questions and offer support. I am indebted to Diko Mihov and the American Foundation for Bulgaria for selecting me and providing me with a generous graduate student fellowship.

Before I started graduate school, I was lucky to have Frank Kelly as my thesis supervisor during my year at Cambridge University. His encouragement and advice on doing research was invaluable in my transition to the world of research. Back in the day, my high-school mathematics teacher Rumyana Karadjova (who taught both me and my long-time friends, classmates and later fellow Theory of Computation graduate students Adriana Karagiozova and Eli Maneva) was a role model who also helped develop my love for mathematics and rigor.

Above all I thank my family and friends. My brother Niki's constant cheer and trust in me, my mother Dobrinka and my father Velin's selfless love and devotion to my happiness and success, have made this work possible and worthwhile. This thesis is dedicated to them.

Table of Contents

1 Introduction **1**

1.1 Part I: Uncertainty, Risk and Optimization 3

1.2 Part II: Measuring Uncertainty through Economics 5

1.3 Part III: Optimization and Economics for Network Routing 7

I Uncertainty, Risk and Optimization:
Bridging Nonconvex and Stochastic Optimization **9**

2 Modeling risk via user utility **11**

2.1 The stochastic optimization model 13

2.2 Optimal solution and optimal target 16

2.3 Optimal solution for a given target 17

2.4 Monotonically increasing costs 25

3 Nonconvex and stochastic optimization with risk **27**

3.1 Approximation technique for nonconvex optimization 28

3.1.1 Building an approximate nonlinear oracle from linear oracle 29

3.1.2 Using the nonlinear oracle to approximate the function optimum 30

3.1.3 Complexity of the nonlinear separation oracle 31

3.2 Approximating stochastic optimization with risk 32

3.3 Extension to approximate linear oracles 37

3.4 Extension to other objectives with exact and approximate linear oracles 39

3.4.1 Probability tail objective 40

3.4.2 Value-at-risk objective 42

3.5 Proofs for the probability tail objective with exact linear oracle 43

3.6 Proofs for the probability tail objective with approximate linear oracle 47

3.7 Discussion 50

4 Low-rank nonconvex optimization **53**

4.1 Rank 2 53

4.2 Rank k: Definitions 57

4.3 Rank k: Analysis 59

4.3.1 The volume of faces of P 60

4.3.2 The volume of faces of the projection 60

4.3.3 The general case 65

4.3.4 Putting the steps together 67

4.4 Approximation algorithms 68

4.5 Efficient algorithms for zonotopes 69

5 General (high-rank) nonconvex optimization **73**

5.1 Correspondence of concave and supermodular optimization 73

5.2 Hardness of approximation of concave minimization 77

5.3 Hardness of approximation of convex maximization 80

6 Applications **83**

6.1 Parametric optimization 83

6.1.1 Parametric shortest path 85

6.2 Multicriteria optimization 89

6.3 Constrained optimization 91

6.4 Stochastic shortest path 92

6.4.1 Exact algorithms 92

6.4.2 Extensions to other distributions 94

6.4.3 Correlations 96

6.4.4 Experimental results 97

II Measuring Uncertainty through Economics:
Mechanisms and Algorithms for Prediction Markets **101**

7 Introduction to prediction markets **103**

8 Prediction markets with a market maker **107**

8.1 Dynamic parimutuel markets 107

8.2 Market scoring rules 109

8.3 Strategic analysis of markets with a market maker 110
8.3.1 Description of the projection market 110
8.3.2 Strategic properties of the projection market 112
8.4 Strategic equivalence of projection market to dynamic parimutuel markets 114
8.5 The projection market vs market scoring rules 117
8.5.1 Strategic equivalence to the spherical market scoring rule 117
8.5.2 Relaxing the restrictions 120
8.5.3 Strategic connections to other market scoring rules 122
8.6 Using the projection-market model for analyzing non-myopic and noisy trade . . . 123
8.6.1 Traders with inertia 123
8.6.2 Analyzing long-range strategies 124
8.7 Concluding remarks on markets with a market maker 125

9 Prediction markets with an auctioneer **127**
9.1 Design goal and challenges 128
9.2 Permutation betting 129
9.2.1 Securities, orders and matching problems 129
9.2.2 Subset betting 131
9.2.3 Pair betting 132
9.3 Complexity of subset betting 133
9.4 Complexity of pair betting 134
9.4.1 Optimal indivisible matching 135
9.4.2 Optimal divisible matching 138
9.4.3 Existence of a match 142
9.5 Concluding remarks on markets with an auctioneer 144

10 Open questions: a market maker or an auctioneer? **147**

III Optimization in Economics:
Mechanisms and Algorithms for Network Routing **149**

11 Motivation **151**

12 First-price auction mechanisms **153**
12.1 Definitions and notation 154
12.2 Deterministic path auction game 155

12.3 Implementation in ϵ-Nash 158
12.3.1 Computing the set of covering flows $\{P_e\}$ 160
12.4 Distribution on demands 160
12.4.1 No equilibrium with 1-parameter bidding 161
12.4.2 Equilibrium with 2-parameter bidding 161
12.4.3 The cooperative game $\mathcal{G}$ 162
12.4.4 Existence of an ϵ-Nash equilibrium 164
12.4.5 Randomized 2-parameter Auction 164

13 Second-price (VCG) mechanism 167
13.1 Definitions and the VCG mechanism 168
13.2 VCG Overpayment in Erdős-Renyi random graphs 170
13.3 Complete graphs with random edge costs 173
13.4 Power-law graphs 174

14 Integrating the VCG mechanism with inter-domain routing 177
14.1 Basic model 178
14.1.1 VCG mechanism 179
14.2 Model with capacities 180
14.3 Model with economic relationships 181
14.4 Distributed computation of VCG payments 182
14.4.1 Basic Model 182
14.4.2 Model with capacities 183
14.4.3 Model with economic relationships 185
14.5 Concluding remarks and open problems 187

15 Appendix: Quasi-concavity background 189
15.1 Concave and Quasi-concave functions 189
15.2 Quasi-convexity of the probability-tail objective 190

Bibliography 191

CHAPTER 1

Introduction

Classical algorithms from theoretical computer science arise time and again in practice. However, practical situations typically do not fit precisely into the traditional theoretical models. Additional necessary components are, for example, uncertainty and economic incentives. Therefore, modern algorithm design is calling for more interdisciplinary approaches, as well as for deeper theoretical understanding, so that the algorithms can apply to more realistic settings and complex systems.

Consider, for instance, the classical shortest path algorithm, which, given a graph with specified edge weights, seeks the path minimizing the total weight from a source to a destination. In practice, the edge weights are often uncertain and it is not even clear what we mean by shortest path anymore: is it the path that minimizes the expected weight? Or its variance, or some another metric? With a risk-averse objective function that takes into account both mean and standard deviation, we run into nonconvex optimization challenges that require new theory beyond classical shortest path algorithm design. Yet another shortest path application, routing of packets in the Internet, needs to further incorporate economic incentives to reflect the various business relationships among the Internet Service Providers that affect the choice of packet routes.

Strategic Algorithms *are algorithms that integrate optimization, uncertainty and economic modeling into algorithm design, with the goal of bringing about new theoretical developments and solving practical applications arising in complex computational-economic systems.*

In short, this thesis contributes new algorithms and their underlying theory at the interface of optimization, uncertainty and economics. Although the interplay of these disciplines is present in various forms in our work, for the sake of presentation, we have divided the material into three categories:

1. In Part I we investigate algorithms at the intersection of *Optimization* and *Uncertainty*. The key conceptual contribution in this part is discovering a novel connection between stochastic and nonconvex optimization. Traditional algorithm design has not taken into account the risk inherent in stochastic optimization problems. We consider natural objectives that incorporate risk, which turn out equivalent to certain nonconvex problems from the realm of continuous optimization. As a result, our work advances the state of art in both stochastic and in nonconvex optimization, presenting new complexity results and proposing general-purpose efficient approximation algorithms, some of which have shown promising practical performance and have been implemented in a real traffic prediction and navigation system.

2. Part II proposes new algorithm and mechanism design at the intersection of *Uncertainty* and

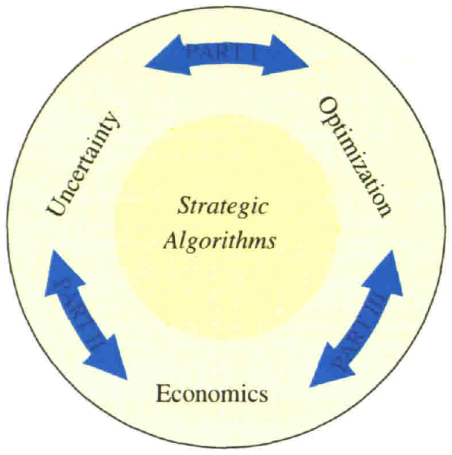


Figure 1.1. Dissertation structure overview.

Economics. In Part I we postulate that the random variables in our models come from given distributions. However, determining those distributions or their parameters is a challenging and fundamental problem in itself. A tool from Economics that has recently gained momentum for measuring the probability distribution of a random variable is an *information* or *prediction* market. Such markets, most popularly known for predicting the outcomes of political elections or other events of interest, have shown remarkable accuracy in practice, though at the same time have left open the theoretical and strategic analysis of current implementations, as well as the need for new and improved designs that handle more complex outcome spaces (probability distribution functions) as opposed to binary or n-ary valued distributions. The contributions of this part include a unified strategic analysis of different prediction market designs that have been implemented in practice. We also offer new market designs for handling exponentially large outcome spaces stemming from ranking or permutation-type outcomes, together with algorithmic and complexity analysis.

3. In Part III we consider the interplay of optimization and economics in the context of network routing. This part is motivated by the network of autonomous systems in the Internet where each portion of the network is controlled by an Internet service provider, namely by a self-interested economic agent. The business incentives do not exist merely in addition to the computer protocols governing the network. Although not currently integrated in those protocols and decided largely via private contracting and negotiations, these economic considerations are a principal factor that determines how packets are routed. And vice versa, the demand and flow of network traffic fundamentally affect provider contracts and prices.

   The contributions of this part are the design and analysis of economic mechanisms for network routing. The mechanisms are based on first- and second-price auctions (the so-called Vickrey-Clarke-Groves, or VCG mechanisms). We first analyze the equilibria and prices resulting from these mechanisms. We then investigate the compatibility of the better understood VCG-mechanisms with the current inter-domain routing protocols, and we demonstrate the critical importance of correct modeling and how it affects the complexity and algorithms necessary to implement the economic mechanisms.

The three parts can be read independently from one another although we view the material as different facets of the same research agenda—modern algorithmic design for complex computational-economic systems. Below we highlight some of the contributions in more detail and give a roadmap of the chapter dependencies within each part.

1.1 Part I: Uncertainty, Risk and Optimization

Imagine driving through uncertain traffic and looking for the optimal route to a destination. What does "optimal" mean? In the absence of uncertainty, one simply finds the shortest path. With uncertainty at hand, stochastic optimization models most commonly look for the feasible solution with smallest expected length. However, a typical user may be averse to the risk inherent in such a solution. Economic models on the other hand postulate that each user has a utility function which captures his degree of risk aversion, and the goal is to optimize expected utility. Such models are integral in portfolio optimization, for example, where, although we want as high a return on our investment as possible, we are also averse to stock depreciation and loss. Therefore, we need to trade off risk and return, which is commonly done by optimizing a linear combination of the mean and standard deviation of the portfolio vector.

Motivated by these economic models, in Chapter 3, we adopt this family of utility functions, parametrized by different convex combinations of the mean and standard deviation of the feasible solution vector. Denote a feasible vector by $\mathbf{x} = (x_1, ..., x_n)$, which is constrained to lie in some feasible set $\mathcal{F} \subset \mathbb{R}^n$, and the vectors of means and variances by $\boldsymbol{\mu}, \boldsymbol{\tau} \in \mathbb{R}^n$ respectively: the mean of a feasible solution would be $\boldsymbol{\mu}^T\mathbf{x}$ and its variance $\boldsymbol{\tau}^T\mathbf{x}$ (where $\boldsymbol{\mu}^T$ denotes the transpose of vector $\boldsymbol{\mu}$), provided the probability distributions for the n coordinates are independent and $x_i = 0$ or 1 for all i. Our problem, then, is to

$$\begin{array}{ll} \text{minimize} & \boldsymbol{\mu}^T\mathbf{x} + k\sqrt{\boldsymbol{\tau}^T\mathbf{x}} \\ \text{subject to} & \mathbf{x} \in \mathcal{F}. \end{array}$$

This objective function is concave and monotone increasing in $\mathbf{x}$, and albeit a function on $\mathbb{R}^n$, it is a low-rank function of rank 2 since it depends only on two dot products, $\boldsymbol{\mu}^T\mathbf{x}$ and $\boldsymbol{\tau}^T\mathbf{x}$. Thus we can project the objective function and feasible set onto the mean-variance plane, $span(\boldsymbol{\mu}, \boldsymbol{\tau})$ and by properties of the objective function the minimum will be attained at an extreme point of the dominant (*i.e.,* the lower-left boundary) of the projected feasible set. With access to an algorithm for solving the deterministic version of the problem, $\{\min \mathbf{w}^T\mathbf{x} \mid \mathbf{x} \in \mathcal{F}\}$ for any given weight vector $\mathbf{w}$, we can enumerate points on the lower left boundary, evaluate the objective function value at each and pick the optimal one.

This gives an exact algorithm for solving the stochastic problem above, whose running time depends on the number of relevant extreme points. To give some examples, in the case of minimum spanning trees and more generally matroids, the algorithm is polynomial; for shortest paths it is subexponential, $n^{O(\log n)}$; however, for most other problems of interest it is exponential. This result is stated in Theorem 6.4.1 for shortest paths and is mentioned in passing in Section 6.1, following from Theorem 6.1.2 and Lemma 6.1.3 for the general framework.

Because of the high parametric complexity and consequently the high number of relevant extreme points for many problems, the exact algorithms may have high running times. Our empirical results from Section 6.4.4 on the other hand show that these algorithms do considerably better in practice than the theoretical average- and worst-case predictions. The intuition is that the number of extreme points is high in very few instances of the problem at hand, whereas most instances only have polynomially many extreme points, resulting in low polynomial average and smoothed complexity (Section 4.1).

Despite the seemingly good practical performance of the exact algorithms described above, it is of both theoretical and practical interest to develop efficient approximation algorithms, with provably low running time and good approximation guarantee. All of Chapter 3 is devoted to developing fully-polynomial approximation schemes, both for the mean-risk objective above, as well as for other objectives involving probability tails:

- maximizing the probability that the feasible solution weight is within a given budget or threshold: $\{\max \mathbf{Pr}(\mathbf{W}^T\mathbf{x} \leq t) \mid x \in \mathcal{F}\}$ in Section 3.4.1, where $\mathbf{W}$ is the vector of random weights;

- the value-at-risk objective $\{\min t \mid \mathbf{Pr}(\mathbf{W}^T\mathbf{x} \leq t) \geq p;\ \mathbf{x} \in \mathcal{F}\}$ in Section 3.4.2, minimizing the threshold t which guarantees us a desired confidence level p that a feasible solution weight will be within that threshold.

For example, in the context of shortest paths, the first problem is to maximize the probability that we arrive on time, namely maximize the probability that the shortest path travel time is within the given deadline. The second problem asks: given a desired 95% confidence level, what is the minimum time we can depart before our deadline so as to arrive on time with at least 95% chance. We give approximation algorithms for the stochastic framework both for the cases when we can solve the underlying deterministic problem exactly (Theorems 3.2.5, 3.4.2) and when we can only approximate it (Theorems 3.3.6, 3.4.3).

The models with specific utility functions above can naturally be relaxed and stated more generally, as minimizing the expected cost of a solution for a general cost (disutility) function, $\{\min \mathbf{E}[C(\mathbf{W}^T\mathbf{x})] \mid \mathbf{x} \in \mathcal{F}\}$, as we do in Chapter 2. Unfortunately, such general formulations are typically intractable, due to multiple challenges of different nature:

(i) Combinatorial difficulty. Even in the absence of randomness, when the edge weights are fully deterministic, a wide class of cost functions reduce to finding the longest path in a graph, which is NP-hard and inapproximable within a reasonable factor [73].

(ii) Distributional difficulty. The distributional assumptions on the element weights W_i may bring a difficulty on their own to the extent that we cannot even compute the distribution of the total weight of a feasible solution $\mathbf{W}^T\mathbf{x} = \sum_{i:x_i=1} W_i$, let alone evaluate the function $\mathbf{E}[C(\mathbf{W}^T\mathbf{x})]$ and minimize it. For example, computing the probability that the sum of n non-identical Bernoulli random variables is less than a given number is #P-hard [80].

(iii) Analytic difficulty. Even with additive distributions such as the Normal distribution, with which we can readily compute the density f of the weight of a feasible solution $\mathbf{W}^T\mathbf{x}$, we might not be able to get a closed analytic form of the objective function $\mathbf{E}[C(\mathbf{W}^T\mathbf{x})] = \int f(u)C(u)du$ and thus cannot optimize it efficiently.

(iv) Functional difficulty. Having computed the distribution of the feasible solution weight $\mathbf{W}^T\mathbf{x}$ and a closed-form expression for the objective function $\mathbf{E}[C(\mathbf{W}^T\mathbf{x})]$, we are left with an integer optimization problem, namely to minimize a (nonlinear or nonconvex) function over the feasible set $\mathcal{F}$, which may be a hard problem, as well.

Not surprisingly in light of these challenges, in Chapter 2 we give hardness of approximation results for the general problem. Nevertheless, we are also able to identify function classes for which we give polynomial and pseudopolynomial algorithms.

We now turn to discussing generalizations in nonconvex optimization, of which the mean-risk objective is a special case. Our main insight in approximating this objective holds for concave minimization in general—the (approximate) optimum lies in the union of half-spaces, in contrast to convex minimization where the optimum is in the intersection of half-spaces. Our technique for the fully polynomial approximation schemes above is also an instance of polytopal approximation of convex bodies, the latter given by the level sets of the objective functions. There is a rich body of research in convex geometry on approximation of convex bodies by polytopes [60], although the emphasis is typically on volume or surface approximation and the convex body specifications are different from our setting, where they are implicitly defined by an objective function. This leaves open the question of how well one can approximate the level sets of an arbitrary function: at present, we need knowledge of the actual function to compute and analyze the complexity of the necessary polytopal approximation. Our approach would extend to other low-rank quasiconcave functions for which we conjecture that the approximating polytopes will have a small (polynomial) number of facets under a mild restriction on the function gradient or on the feasible set.

Independently, in Chapter 4, we provide average and smoothed bounds for the number of extreme points on a polytope projected on a subspace of small dimension. These bounds have two important implications: first, a polynomial smoothed bound provides a theoretical justification that a high number of extreme points occurs very rarely and thus, like the simplex method for linear programming, our exact algorithms may actually have good practical performance. Second, the smoothed bounds can be formally turned into an approximation algorithm by running the exact algorithm multiple times for a small number of steps over perturbed values of the input parameters. With this, we provide alternative approximation algorithms (although the approximation here is additive, the algorithms are randomized and the running time is higher) to the stochastic optimization framework that motivated us to look at low-rank nonconvex optimization problems.

Part of the material presented in Chapters 2, 4, 5 and 6.4 has appeared in publications joint with Matthew Brand, David Karger, Jonathan Kelner and Michael Mitzenmacher [98, 99, 78].

1.2 Part II: Measuring Uncertainty through Economics

A key component in models involving uncertainty is the random variables distribution, often assumed to be known or having known parameters. In this section we provide new mechanisms and algorithms for an increasingly common and accurate method of estimating uncertainty and its parameters: prediction markets, also known as information or decision markets.

Prediction markets are structured similarly to financial markets, except the main purpose is to aggregate traders' information so as to predict the outcome of an event. It has long been observed that trading reveals information. Buying or selling a financial security in effect is a wager on the security's value. For example, buying a stock is a bet that the stock's value is greater than its current price. Each trader evaluates his expected profit to decide the quantity to buy or sell according to his own information and subjective probability assessment. The collective interaction of all bets leads to an equilibrium that reflects an aggregation of all traders' information and beliefs. In practice, this aggregate market assessment of the security's value is often more accurate than other forecasts relying on experts, polls, or statistical inference [108, 109, 44, 8, 107].

A lot of what we know about prediction markets accuracy and results stems from empirical data about existing prediction markets such as the Iowa Electronic Markets, TradeSports, the Hollywood Stock Exchange, etc. Since the market designs and trading behavior are very complex to capture in realistic theoretical models, the field of prediction markets leaves open fundamental questions such as what are optimal prediction market designs and what is the optimal strategic behavior.

In Part II we present strategic analysis of prediction markets that gives a unified theoretical approach to understanding their properties, as well as optimal trading behavior. We then propose and analyze several new market designs for handling exponentially many permutation or ranking-type outcomes.

Prediction market designs can be divided into two main categories. In one category (Chapter 9) they operate as an *auction* in which orders on the different securities are matched risklessly by a book-keeper. This market structure leads to a complex price formation process in which it may be difficult to infer resulting prices and the corresponding probabilities of the outcomes. Another challenge is that we may not be able to find matching traders who agree on a price and trade would not occur—the so-called "thin markets" problem. This thesis presents new designs for handling exponentially large outcome spaces resulting from permutation or ranking-type outcomes where the event one is trying to predict is the result of a tournament or a competition of n candidates. Our work is similar in spirit to that of Fortnow *et al.* [45] who instead consider prediction market designs for events whose outcomes can be expressed as logical formulas, and prove hardness results for the auctioneer problem in that setting. With a large number of outcomes it is impractical to have a security representing each possible outcome: we need to restrict the number and types of securities, which means restricting the type of information or beliefs the traders can express. For example, if we only had n securities for the n possible winners in the tournament, we would lose as a participant a trader who does not have information on the winner, but instead has some belief about the ranking of a subset of the candidates (*e.g.,* candidate A would do better than candidate B): this trader would not be able to express such belief by trading with only these n securities. On the other hand, a higher number of securities and higher expressivity may lead to computationally intractable problems of finding matching traders who can trade.

Somewhat surprisingly, we show that in this perceived trade-off between expressivity and tractability, these two goals for practical prediction market design are not necessarily conflicting. With one family of securities, in which each security corresponds to a pair of ranked candidates, we both have a limited expressivity of only $O(n^2)$ possible securities (one for each ordered pair), and the problem of identifying matching traders is NP-hard. With another family of securities where traders can bid on which subset of candidates would place in a given position or which subset of positions a candidate would end up in, we have exponentially many possible securities and at the same time we can solve the auctioneer problem—to identify matching traders—in polynomial time. Curiously, the economic analysis of the auctioneer's revenue maximization and matching problems is based on classical results from combinatorial optimization such as the NP-hardness of the minimum feedback arc problem and the polynomial-time solvability maximum matching.

In the second category of prediction markets (considered in Chapter 8), the challenges of thin markets and complex price formation are alleviated by having a deterministic price-setting rule based on trading volume and a centralized *market maker* who is always willing to trade at the given price. A drawback in this type of market is that the market maker may incur a loss by fulfilling all orders at the current price. Nevertheless, such loss is not necessarily a negative, as long as it is bounded, when viewed as a subsidy or price that the market maker incurs in order to aggregate and acquire information. Despite the growing popularity and implementation of prediction market designs with a market maker, their strategic and theoretical properties are not fully understood. For example, the dynamic parimutuel market proposed by Pennock [106] was implemented in the Yahoo! Buzz Market and a previous version crashed after traders found and exploited arbitrage opportunities [86], calling for a change in the market design. In this chapter, we propose an abstract prediction market, and show how to use it as a tool for analyzing the strategic properties of existing market designs, as well as for guiding future designs. In particular, we prove that our prediction market is strategically equivalent to existing market mechanisms and thus its analysis automatically transfers to these mechanisms. The benefit of using our abstract market model is that it has an attractive geometric visualization which makes the analysis and strategy computation significantly easier than in the market designs operating in practice.

The material in Chapter 8 is based on joint work with Rahul Sami [100] and Chapter 9 is based on joint work with Yiling Chen, Lance Fortnow and David Pennock [20].

1.3 Part III: Optimization and Economics for Network Routing

This part introduces and analyzes new mechanisms and algorithms at the interface of *Economics* and *Optimization*. As mentioned in the motivating example at the beginning of this introduction, when packets are routed in a network, the business relationships between the different Internet Service Providers form an integral part in the decisions what routes the packets should take. Therefore, economic modeling needs to be integrated into the shortest path algorithms; in particular, we need to examine mechanisms which determine both what route a packet should take and what price the packet should pay for being forwarded by the different providers. Modeling the Internet and the economic relationships that govern it is a complex problem in itself, and the models critically influence the optimization problems and resulting algorithms for computing the routes and prices. Some of the most basic models assume that different edges (or nodes) in the network are owned by different self-interested agents who incur a cost for routing packets through and bid a transit price in an auction-type environment. After the edge price declarations, the packet typically picks the lowest-price path and pays each edge along the path according to the mechanism specifications. The contributions in Part III of this thesis include an analysis of the resulting prices from the celebrated Vickrey-Clarke-Groves truthful mechanism in which each edge on the cheapest path is paid the most it could have bid and still remained on the cheapest path. This mechanism is known to result in very high overpayments in which the packet ends up paying the selected route a price much higher than the cost of the next best alternative; in Chapter 13 we show that in more realistic network models this overpayment is insignificant. We also consider first-price mechanisms in which edges are paid their bid, and show that the route payments under those mechanisms are much lower than the corresponding VCG payments (Chapter 12). In the last Chapter 14, we extend this basic model to a more realistic one of autonomous systems as nodes in a graph, each one having different values for the incoming and outgoing traffic along its adjacent edges. We show that depending on how we incorporate the economic relationships into the model, the VCG allocation and price computations vary from efficiently computable in a distributed fashion to incompatible with the current Border Gateway Protocol which governs the routing decisions at the Autonomous Systems level of the Internet.

Chapter 12 is joint work with Nicole Immorlica, David Karger, Rahul Sami [69]; Chapter 13 was done jointly with David Karger [74, 75] and Chapter 14 was the result of a collaboration with Alexander Hall and Christos Papadimitriou [63].

Part I

Uncertainty, Risk and Optimization: *Bridging Nonconvex and Stochastic Optimization*

CHAPTER 2

Modeling risk via user utility

Consider the general combinatorial problem that minimizes the weight of a feasible solution:

$$\begin{aligned} \text{minimize} \quad & \mathrm{w}^T\mathrm{x} \\ \text{subject to} \quad & \mathrm{x} \in \mathcal{F}. \end{aligned} \tag{2.1}$$

Suppose now that the weight vector $\mathbf{W}$ is stochastic and we need to find an optimal stochastic solution. But what does one mean by optimal solution in this case? Is it the solution minimizing expectation, or perhaps variance, or a combination of the two? In real applications the user is typically averse to risk. Therefore we need a model that incorporates this risk.

This chapter introduces a general approach for modeling risk via a user-dependent utility function. We present new complexity results and efficient algorithms for the resulting stochastic problems. An original motivation for this area came from route planning in road networks. Current navigation systems use information about road lengths and speed limits to compute deterministic shortest or fastest paths. When realized (driven), these paths often turn out to be quite suboptimal, for the simple reason that the deterministic solution ignores the inherent stochasticity of traffic, as well as changing parameters of the stochastic traffic distributions. The statistics of traffic flows are now estimable in real time from road sensor networks; thus, we ask how effectively and efficiently such information can be exploited. The same questions hold for many other combinatorial optimization problems which face uncertainty in real applications.

For a stochastic combinatorial problem we need to optimize an objective that makes some trade-off between the expected weight (mean) of the solution and its reliability (variance). Optimizing one or the other, though tractable algorithmically, makes little sense. For example, finding the solution with the lowest expected travel time has little value because a user can only sample a single realization of that solution in the given uncertain environment; with variance unoptimized, that realization could be quite far from the mean. Optimizing a linear combination of the mean and variance is another possibility, though this seems ad-hoc and not clearly motivated. Interestingly it turns out to be a special case of our formulation.

Decision theory, the standard framework for making optimal plans and policies under uncertainty, expresses the trade-off between expected weight and reliability through a utility or cost function $C : \mathbb{R} \to \mathbb{R}^+$. In our setting $C_t(\mathrm{w}^T\mathrm{x})$ assesses a penalty for obtaining a solution with weight $\mathrm{w}^T\mathrm{x}$ that may depend on some threshold parameter t. For example, a linear cost function $C(.)$ minimizes the expected solution weight; quadratic cost minimizes variance; the minimizer of their weighted sum takes a surprising form related to the cumulant generating function of the solution weight distribution.

Unfortunately, the general formulation $\{\min \mathbf{E}[C(\mathbf{W}^T\mathbf{x})] \mid \mathbf{x} \in \mathcal{F}\}$ is not only hard from a computational complexity point of view but faces obstacles of different nature:

- Combinatorial difficulty. Even in the absence of randomness, when the edge weights are fully deterministic, a wide class of cost functions reduce to finding the longest path in a graph, which is NP-hard and inapproximable within a reasonable factor [73].

- Distributional difficulty. The distributional assumptions on the element weights W_i may bring a difficulty on their own, to the extent that we cannot even compute the distribution of the total weight of a feasible solution $\mathbf{W}^T\mathbf{x} = \sum_{i:x_i=1} W_i$, let alone evaluate the function $\mathbf{E}[C(\mathbf{W}^T\mathbf{x})]$ and minimize it. For example, computing the probability that the sum of n non-identical Bernoulli random variables is less than a given number is #P-hard [80].

- Analytic difficulty. Even with additive distributions such as the Normal distribution, with which we can readily compute the density f of the weight of a feasible solution $\mathbf{W}^T\mathbf{x}$, we might not be able to get a closed analytic form of the objective function $\mathbf{E}[C(\mathbf{W}^T\mathbf{x})] = \int f(u)C(u)du$ and cannot optimize it efficiently. [1]

- Functional difficulty. Having computed the distribution of the feasible solution weight $\mathbf{W}^T\mathbf{x}$ and a closed form expression for the objective function $\mathbf{E}[C(\mathbf{W}^T\mathbf{x})]$, we are left with an integer optimization problem, to minimize a (nonlinear or nonconvex) function over the feasible set $\mathcal{F}$, which may be a hard problem, as well.

Not surprisingly in light of these challenges, in this chapter our contributions include hardness of approximation results for the general problem. Nevertheless, we are also able to identify function classes for which we give polynomial and pseudopolynomial algorithms.

The stochastic problems we consider fall into two main categories: "What is the optimal solution for a given threshold parameter?" and "What is the optimal choice of threshold parameter and solution which lead to a lowest overall cost?" (for example, in the case of shortest paths where the threshold parameter is a deadline to reach the destination, these questions ask "What is the optimal time to depart and optimal route to take?" and "Now that I am on the road, what is the optimal route for that deadline?"). Surprisingly, for some cost functions of interest, the former question is tractable while the latter is NP-hard.

This highlights the dependence of stochastic solutions on the threshold parameter. For example, imagine that we have a choice of two solutions and only care to pick one whose weight does not exceed the threshold. Maximizing the probability of doing so implies that $C(.)$ is a step function. If we have a low threshold, a costlier and highly *variable* solution will actually be preferable to a cheaper and highly reliable one because the less predictable solution offers a greater chance of remaining below the threshold (see Figure 2.1).

[1]This is a common problem in decision theory and related fields, which therefore focus attention on conjugate pairs of function and distribution families (more precisely, conjugate priors), *i.e.*, function-distribution pairs for which the integral can be computed in a closed form and the expected cost function lies in the same family as the original cost function $C(.)$.

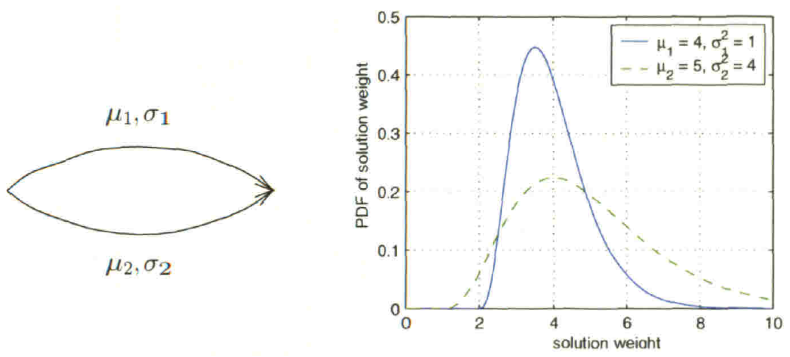


Figure 2.1. Suppose we have a choice of two solutions with mean and variance $\mu_1 = 4, \tau_1 = 1$ and $\mu_1 = 5, \tau_1 = 4$ respectively. The optimal solution depends on the cost threshold. Solution one is on average cheaper ($\mu_1 < \mu_2$) and more reliable ($\tau_1 < \tau_2$), but for a threshold less than 2, solution two offers a higher (albeit small) probability of being below the threshold.

2.1 The stochastic optimization model

Let $\mathcal{U} = \{e_1, ..., e_n\}$ be a ground set of elements. The elements have independent stochastic weights W_i specified by probability density functions $f_i(.)$ with mean μ_i and variance τ_i. We have a set of feasible subsets or solutions $\mathcal{F}$ and are interested in finding a solution that optimizes the user's expected utility. We specify the latter via a cost function $C(\cdot)$, thus we seek to solve

$$\begin{array}{ll} \text{minimize} & \mathbf{E}[C(\mathbf{W}^T\mathbf{x})] \\ \text{subject to} & \mathbf{x} \in \mathcal{F}, \end{array} \tag{2.2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the indicator vector for a solution and $\mathbf{W} = (W_1, ..., W_n)$ is the vector of stochastic weights. Denote the mean and variance vectors of $\mathbf{W}$ by $\boldsymbol{\mu}$ and $\boldsymbol{\tau}$ respectively.

The cost function may depend on a target parameter—for example, a budget or deadline that we do not want the stochastic solution weight to exceed. Then, the cost function values can be interpreted as penalties for being below or above this target. (For example, this can be the available time we have to reach our destination in a route planning setting, or the weight limit of items we can fit in a suitcase, etc.) By independence, the distribution of the stochastic solution weight $\mathbf{W}^T\mathbf{x}$ has mean $\boldsymbol{\mu}^T\mathbf{x}$ and variance $\boldsymbol{\tau}^T\mathbf{x}$. Denote its probability density function by $f_{\mathbf{W}^T\mathbf{x}}(.)$. Then the expected cost of a solution weight is

$$\mathbf{E}[C(\mathbf{W}^T\mathbf{x}, t)] = \int_{w_1,...,w_n} \Big[f_{\mathbf{W}^T\mathbf{x}}(w_1, ..., w_n)C(\mathbf{W}^T\mathbf{x}, t)\Big] dw_1...dw_n. \tag{2.3}$$

We now distinguish two different problems:

1. Find the optimal solution $\mathbf{x}$ and optimal target t:

$$\min_{t, \mathbf{x} \in \mathcal{F}} \mathbf{E}[C(\mathbf{W}^T\mathbf{x}, t)] \tag{2.4}$$

2. Find the optimal solution for a given target.

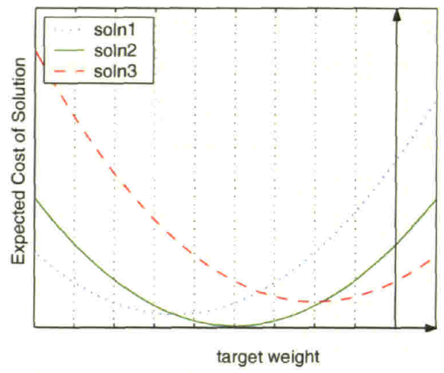


Figure 2.2. Each solution has an associated expected cost as a function of the target weight t. For the target weight marked by the vertical arrow, solution 3 is optimal; however, the globally optimal target is located at the minimum of the solution 2 cost curve.

If we graph the expected cost of each feasible solution as a function of the target weight t, we obtain a family of curves, cartooned in Figure 2.2. The best solution for a given target is indicated by the lowest curve at that point. Note that each solution may be optimal over a different range of targets. The global miminum of the lower envelope of all such curves indexes the optimal target.

Calculating the cost of a single solution

In general, the expected cost expression in equation (2.3) may be impossible to compute in a closed form. We will therefore focus on two families of cost functions for which the integral can be computed exactly, and which are a sensible model of user utility: polynomials and exponentials. In this section we assume that the user values her resources and does not want a solution that is too much below or above the target. (In the following section, we will consider cost functions that do not penalize for being below target.) Thus the cost function should be expressive enough to (asymmetrically) penalize being both below and above target. Although many of the results of subsequent sections apply to general polynomial functions (and our hardness results hold for arbitrary functions with global minima), we will consider here quadratic and quadratic+exponential cost functions for illustrative purposes.

Quadratic Cost Suppose the penalty for being above or below the target is quadratic in the amount we deviate from the target, that is $C(\mathbf{W}^T\mathbf{x}, t) = (t - \mathbf{W}^T\mathbf{x})^2$. Then the expected cost of a solution is

$$\begin{aligned}
\mathbf{E}[C(\mathbf{W}^T\mathbf{x}, t)] &= \mathbf{E}[(t - \mathbf{W}^T\mathbf{x})^2] = t^2 - 2t\,\mathbf{E}[\mathbf{W}^T\mathbf{x}] + \mathbf{E}[(\mathbf{W}^T\mathbf{x})^2] \\
&= t^2 - 2t\boldsymbol{\mu}^T\mathbf{x} + (\boldsymbol{\mu}^T\mathbf{x} + \boldsymbol{\tau}^T\mathbf{x}) \\
&= (t - \boldsymbol{\mu}^T\mathbf{x})^2 + \boldsymbol{\tau}^T\mathbf{x}.
\end{aligned}$$

We see that this expected cost is always nonnegative. Thus if we find a solution with zero variance and mean equal to the target, this would be an optimal solution.

The quadratic cost function might not be regarded as realistic since it assigns the same penalty to being equally below and above the target. As we saw above, this leads to preferring the most certain solution, without any care for its average weight. On the other hand, linear costs, which favor on average lower-weight solutions, do not have any effect when added to the quadratic cost other than shifting the effective target. Thus, we augment the quadratic cost function with an exponential term which gives a higher penalty to being above the target.

Quadratic+Exponential Cost Consider cost function $C(y) = y^2 + \lambda e^{ky}$, where again y is the deviation from the target weight and $\lambda \geq 0$ and k are parameters which determine the strength of the penalty for exceeding the target. The sign of k can be negative if one is more averse to a solution weight below the threshold than exceeding the threshold.

In this case, we can still get a closed form expression for the expected cost of a solution.

$$\begin{aligned}\mathbf{E}[C(\mathbf{W}^T\mathbf{x} - t)] &= \mathbf{E}[(\mathbf{W}^T\mathbf{x} - t)^2 + \lambda e^{k(\mathbf{W}^T\mathbf{x}-t)}] \\ &= (t - \boldsymbol{\mu}^T\mathbf{x})^2 + \boldsymbol{\tau}^T\mathbf{x} + \lambda e^{-kt}\,\mathbf{E}[e^{k\mathbf{W}^T\mathbf{x}}] \\ &= (t - \boldsymbol{\mu}^T\mathbf{x})^2 + \boldsymbol{\tau}^T\mathbf{x} + \lambda e^{-kt}\prod_{i=1}^{n}\mathbf{E}[e^{kW_i x_i}].\end{aligned}$$

Choice of weight distributions

It is natural to model the stochastic weights by normal distributions. For a normally distributed random variable $W \sim N(\mu, \tau)$, we have $\mathbf{E}[e^{kW}] = e^{(k\mu + k^2\tau/2)}$. Thus, the expected cost of a solution x with independent normally distributed weights W_i of its components is given by

$$\mathbf{E}[C(\mathbf{W}^T\mathbf{x} - t)] = (t - \boldsymbol{\mu}^T\mathbf{x})^2 + \boldsymbol{\tau}^T\mathbf{x} + \lambda e^{-kt} e^{k\sum_i(\mu_i x_i + k\tau_i x_i^2/2)}. \tag{2.5}$$

For the special case of discrete combinatorial problems where $x_i = 0$ or 1, we can rewrite the last term as follows:

$$\mathbf{E}[C(\mathbf{W}^T\mathbf{x} - t)] = (t - \boldsymbol{\mu}^T\mathbf{x})^2 + \boldsymbol{\tau}^T\mathbf{x} + \lambda e^{-kt} e^{k(\boldsymbol{\mu}^T\mathbf{x} + k\boldsymbol{\tau}^T\mathbf{x}/2)}. \tag{2.6}$$

However, the normal distribution is sometimes unrealistic as it assigns positive probability to negative weights. A more physically appropriate distribution which does not have probability mass on the negative axis, is the Gamma distribution [88]. Gamma distributions are also proposed in Fan *et al.* [37].

We write $W_i \sim \gamma(a_i, b_i)$ to denote a Gamma distributed weight with shape parameter a_i and width parameter b_i. The mean of this distribution is given by $\mu_i = a_i b_i$ and the variance is $\tau_i^2 = a_i b_i^2$. The density of the gamma distribution is given by

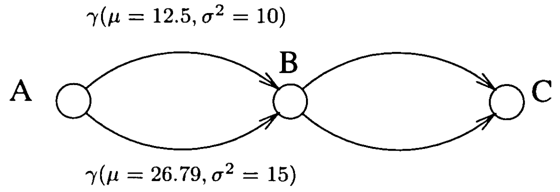
$$\gamma(a, b, w) = \frac{w^{a-1}e^{-w/b}}{b^a\Gamma(a)},$$

where $\Gamma(a) = \int_0^\infty y^{a-1}e^{-y}dy$ is the gamma function. The Gamma distribution has strictly nonnegative support and we can additionally specify a minimum weight by shifting w. To keep notation uncluttered, we will use unshifted distributions; the generalization to shifts is straightforward.

For a gamma random variable $W \sim \gamma(a, b)$ we have $\mathbf{E}[e^{kW}] = (1 - kb)^{-a}$. Therefore, the expected cost of a solution with independent gamma distributed weight components is given by:

$$\left(t + \boldsymbol{\mu}^T\mathbf{x}\right)^2 + \boldsymbol{\tau}^T\mathbf{x} + \lambda e^{-kt}\left[\prod_{i=1}^{n}(1 - kx_i b_i)^{-a_i}\right], \tag{2.7}$$

which no longer has a simple analytic expression for the minimizing target t.


$\gamma(\mu = 12.5, \sigma^2 = 10)$
A
B
C
$\gamma(\mu = 26.79, \sigma^2 = 15)$

Path	$A \to B$	$A \to C$
Top edges	$(22.2, 123.1)$	$(46.5, 526.7)$
Bottom & Top	-	$(60.7, 524.6)$
Bottom edges	$(36.3, 125.0)$	$(74.8, 522.7)$

Table 2.1. In the graph above, the feasible solutions are represented by paths from A to C. The top edges are identically Gamma distributed, and so are the bottom edges. When we use the quadratic+exponential cost function $C(y) = y^2 + e^y$, the optimal path from A to C uses the two bottom edges while the optimal subpaths from A to B and from B to C use the top edges. The table entries give the values of the optimal target and expected cost at the minimum for each path.

2.2 Optimal solution and optimal target

In this section we consider the subproblem of jointly optimizing for the solution and target weight. We show that the quadratic cost function with general edge distributions, as well as the quadratic+ exponential cost with Gaussian distributions, result in selecting the lowest variance solution. Therefore this problem variant can be solved via an algorithm for the corresponding deterministic optimization problem (2.1), taking variances as the deterministic component weights. On the other hand, the quadratic+ exponential cost with Gamma travel distributions does not satisfy a suboptimality property (a subset of an optimal solution is not optimal for the corresponding subproblem), and remains an open problem even for easy deterministic combinatorial problems that can be solved via dynamic programming. In essence, the stochastic layer changes the combinatorial structure of the problem and there is no clear reduction from the stochastic to the deterministic version of the problem.

Recall that when the cost is quadratic, $C(y) = y^2$, the expected cost of a given solution is minimized at target $t = \mu^T \mathrm{x}$, that is the average weight of the solution, and at this optimum, the expected cost is the variance of the solution, $EC_{min} = \tau^T \mathrm{x}$. Therefore, we can find the optimal solution—the one of smallest total variance, with an application of the underlying deterministic problem where the deterministic weights are set to the variances τ_i. Consequently, the optimal target would be given by the mean weight of that solution. Thus, the optimal target problem turns out easy in the case of a quadratic cost. The second problem of finding the optimal solution for a given target does not benefit from the simple form of the expected cost function; we show in the following section that it is NP-hard.

If we add an exponential penalty for being late by taking $C(y) = y^2 + \lambda e^{ky}$, the expected cost for a path under Gamma distributions still has a simple closed form, given by Equation (2.7). However, we lose the separability property of the quadratic cost functions, which allows one to reduce the problem to its deterministic version. (That is, we cannot split the minimum cost value EC_{min} as a sum of the costs of individual components of the solution.)

Theorem 2.2.1. *Finding the optimal solution and optimal target for the stochastic combinatorial problem (2.4)*

(i) under quadratic cost and general distributions can be solved exactly with an algorithm for solving the underlying deterministic problem (2.1).

(ii) under quadratic+exponential cost and normal distributions can be solved exactly with an algorithm for solving the underlying deterministic (discrete combinatorial) problem.

(iii) under quadratic+exponential cost and general distributions, does not satisfy the suboptimality property (namely a subset of the optimal solution may not be optimal for the corresponding subproblem).

Proof. Part (i) follows from the discussion above. We show part (iii) via a counterexample in which suboptimality does not hold for any subproblem. Consider a set of feasible solutions which can be represented as paths in a graph. In particular, take the graph with two parallel pairs of edges in Table 1. All edges are Gamma-distributed with mean-variance pairs $(12.5, 10)$ on the top and $(26.8, 15)$ on the bottom. The optimal path from A to C consists of the lower two edges with optimal target traversal time of 74.79 units (*e.g.*, which can be thought of as optimal departure time before a given deadline) and minimum cost 522.65. However, the best paths from A to B and from B to C both consist of the top edges with target times 22.18 and minimum cost 123.14. Thus, no subpath of the optimal path from A to C is optimal.

Curiously, the same cost function with normally distributed edge weights admits a reduction to the underlying deterministic problem, when the latter is a discrete combinatorial problem, *i.e.*, $x_i = 0$ or 1. In part (ii), the expected cost of a solution given by Equation (2.5), can be re-written as

$$y^2 + s + e^{ky}e^{k^2 s/2}, \tag{2.8}$$

after the change of variables $y = t - \boldsymbol{\mu}^T \mathrm{x}$. In particular, a path with a higher total variance will have an expected cost function strictly above that of a path with a lower variance, because for $s_1 < s_2$, $k \neq 0$ and for any fixed y, $y^2 + s_1 + e^{ky}e^{k^2 s_1/2} < y^2 + s_2 + e^{ky}e^{k^2 s_2/2}$. Hence the path of lowest variance will have the lowest minimum expected cost, and we can find it with any available algorithm for the underlying deterministic problem, with weights equal to the variances. Thus, when weights are normally distributed, both the quadratic and quadratic plus exponential cost functions will choose the same solution, although the optimal target weight would naturally be bigger under the second family of cost functions. (Intuitively, in the shortest path setting, since we are much more averse to being late, we would be prone to depart earlier and give ourselves a bigger time window for reaching the destination.) $\square$

Remark 2.2.2. *The last part of Theorem 2.2.1 suggests that the stochastic problem in that setting cannot be solved by a reduction to the underlying deterministic problem, in contrast to the remaining problem variants.*

The problem of finding the optimal solution for a given target weight is again NP-hard, as in the quadratic cost case.

2.3 Optimal solution for a given target

From here until the end of the section we will focus on route planning under uncertainty, namely the feasible set $\mathcal{F}$ consists of all ST-paths, for a given source S and destination T in a graph.

We first prove NP-hardness results, and then give pseudopolynomial algorithms for the quadratic and quadratic+exponential cost functions, which generalize to polynomial (plus exponential) cost functions.

In the route planning example, we may be interested in the optimal solution and optimal departure time to a destination, while planning ahead of time. Once we start our journey, it is natural to ask for an update given that current traffic conditions may have changed. Now, we are really posing a new problem: to find the solution of lowest expected cost, for a given target parameter t. This may sound like a simpler question than the one of jointly finding the optimal solution and optimal target. But it turns out to be NP-hard for a very broad class of cost functions.

Complexity of costs with global minimum

In this section, we show NP-hardness and hardness of approximation results for arbitrary cost functions with global minima, when we are looking for the optimal solution with a given target.

Let $C(t)$, the penalty for arriving at the destination at time t, be any function with a global minimum at t_{min}. In case of several global minima, let t_{min} be the smallest one. Denote the number of nodes in the graph by n.

Theorem 2.3.1. *The problem of finding a lowest-cost simple ST-path is NP-hard.*

Proof. Suppose all edges have deterministic unit edge lengths. Then the cost of departing at time t along a path with total length L is simply $C(t + L)$.

Consider departure time $t = t_{min} - (n - n^\epsilon)$. If there exists a path of length $n - n^\epsilon$, it would be optimal since its cost would be

$$C(t + n - n^\epsilon) = C(t_{min}) \leq C(t + L) \tag{2.9}$$

for all other paths of any length L. In particular, since t_{min} is the leftmost global minimum, we have a strict inequality for paths of length $L < n - n^\epsilon$. Now suppose the optimal path is of length L^*. We have three possibilities:

1. $L^* < n - n^\epsilon$. Then by above, there is no path of length $n - n^\epsilon$.

2. $L^* = n - n^\epsilon$. Then we have found a path of length $n - n^\epsilon$.

3. $L^* > n - n^\epsilon$. Then by removing edges, we can obtain a path of length exactly $n - n^\epsilon$.

Therefore, the problem of finding an optimal path reduces to the problem of finding a path of length $n - n^\epsilon$ where $\epsilon < 1$. Since the latter problem is NP-complete [73], our problem is NP-hard. □

Intuitively, if we incur a higher cost for earlier arrivals and depart early enough, the problem of finding an optimal path becomes equivalent to the problem of finding the longest path. Further, if the cost function is not too flat on the left of its minimum, we can see that an approximation of the min-cost path automatically gives a corresponding approximation of the longest path; hence a corollary to the above is that the optimal path is hard to approximate.

Corollary 2.3.2. *For any cost function which is strictly decreasing and positive with slope of absolute value at least $\lambda > 0$ on an interval $[-\infty, t_{min}]$, there does not exist a polynomial constant factor approximation algorithm for finding a simple path of lowest expected cost at a given departure time prior to t_{min}, unless $P = NP$.*

Proof. Suppose the contrary, namely that we can find a path of cost $C = (1+\alpha)C_{opt}$ where C_{opt} is the cost of the optimal path, and $\alpha > 0$ is a constant.

Assume as in the theorem above that we have an n vertex graph with unit length edges and consider departure time $t = t_{min} - (n-1)$ at the source. Let L_{opt} be the length of the optimal path and let L be the length of the path that we find. Then $L \leq L_{opt}$ so L_{opt} is the longest path between the source and the destination and $\frac{C - C_{opt}}{L_{opt} - L} \geq \lambda$, otherwise there would be a point in $[-\infty, t_{min}]$ of absolute slope less than λ. Hence $L_{opt} - L \leq \lambda\alpha C_{opt}$ and so $L_{opt}/L \leq 1 + (\lambda\alpha C_{opt}/L) \leq 1 + \lambda\alpha C_{opt}$, where $C_{opt} = C(t_{min})$ is constant, so this would give a polynomial constant factor approximation algorithm for the longest path problem, which does not exist unless $P = NP$ [73]. □

Remark 2.3.3. *We can obtain a stronger inapproximability result based on the fact that finding a path of length $n - n^{\epsilon}$ is NP-complete, for any $\epsilon < 1$ [73]. However, our goal is simply to show the connection between the inapproximability of our problem to that of the longest path problem and show the need to settle for non-simple paths in the algorithms of the following section.*

The NP-hardness result in Theorem 2.3.1 crucially relies on simple paths, and it makes optimal paths equivalent to longest paths (due to a very early departure time), which is not usually the case. We can show that finding optimal paths is NP-hard even for more reasonable start times and non-simple paths, via a reduction from the subset-sum problem, for any cost function with a unique global minimum.

Theorem 2.3.4. *Suppose we have a cost function $C(\tilde{t})$ with a unique minimum at t_{min}, which gives the penalty of arriving at the destination at time $\tilde{t}$. Then for a start time t at the source, it is NP-complete to determine if there is a path P to the destination of expected cost $EC_P(t) \leq K$, where*

$$EC_P(t) = \int_0^{\infty} f_P(y)C(t+y)dy,$$

and $f_P(y)$ is the travel time distribution on path P.

Proof. The problem is in NP since there is a short certificate for it, given by the actual path if it exists.

To show NP-hardness, we reduce from the Subset Sum problem, namely given a set of integers $\{w_1, ..., w_n\}$ and a target integer t, is there a subset which sums exactly to t? The subset sum problem, which is a special case of the knapsack problem, is NP-complete [22]. Set $K = C(t_{min})$. Consider the graph in Figure 2.3 with deterministic edge travel times $w_1, ..., w_n$ on the bottom and 0 on the top. Any path P from S to T in this graph has travel time $\sum_{i \in P} w_i$ and cost $C(t' + \sum_{i\in P} w_i)$ if we leave the source at time $t' = t_{min} - t$. Since the cost function $C(t)$ has a unique global minimum at t_{min}, there is a path of cost at most $K = C(t_{min})$ if and only if there is a path with travel time satisfying $t' + \sum_{i \in P} w_i = t_{min}$, *i.e.*, if and only if there is a subset of the w_i's summing exactly to t. □

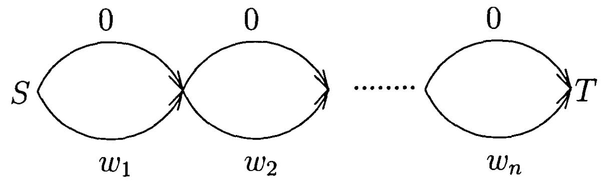


Figure 2.3. If we can solve for the optimal path with a given start time t in this graph, then we can solve the subset sum problem with integers $w_1, ..., w_n$ and target t.

Remark 2.3.5. *Note that Theorem 2.3.1 only shows that it is NP-hard to find a simple optimal path. Theorem 2.3.4 on the other hand applies to non-simple paths as well, since the subset sum problem is NP-complete even if it allows for repetitions [22].*

Complexity of stochastic weights

The theorems in the preceding section show that the stochastic routing problem contains instances with deterministic subgraphs that make routing NP-hard, though we do not know whether the class of purely stochastic routing problems (with non-zero variances) is also NP-hard with general cost objectives and travel time distributions. Indeed, there are known problems in scheduling where the scenario with deterministic processing times is NP-hard, while its variant with stochastic, exponentially distributed processing times can be solved in polynomial time [18].

It may be difficult to extend Theorems 2.3.1 and 2.3.4 to the non-zero variance case partly because the integral defining the expected cost will likely not have a closed form for most cost functions. However, we can prove NP-hardness similarly to Theorem 2.3.4 for stochastic instances and the function classes we considered earlier, where we know the form of the expected costs functions.

Recall that, under quadratic cost, the expected cost of departing at time t along a path P with general edge travel time distributions is $\left(t + \boldsymbol{\mu}^T \mathrm{x}\right)^2 + \boldsymbol{\tau}^T \mathrm{x}$. Define `Stochastic Cost Routing` to be the problem of deciding whether, for a fixed departure time t, there is a path of expected cost less than K for some constant K.

Corollary 2.3.6. `Stochastic Cost Routing` *is NP-hard for quadratic cost with general edge distributions and quadratic+exponential cost with Gamma distributions.*

Proof. In both cases the proof reduces to that of Theorem 2.3.4 by choosing the means and variances on the top and bottom edges carefully so that the parts of the expected cost function which contain the variances become equal for each path.

For the quadratic cost case, this is straightforward by choosing the same variance τ_i^2 to each pair $i = 1, ..., n$ of top and bottom edges in Figure 2.3. The quadratic+exponential cost with Gamma travel times takes a little more work since the means and variances are not so well separated in the exponential term. $\square$

We are going to show that finding optimal solutions is NP-hard via a reduction from the subset-sum problem, for any cost function with a unique global minimum.

Theorem 2.3.7. *Suppose we have a cost function $C(y)$ with a unique minimum at y_{min}, which gives a penalty for exceeding the target by y units. Then for a given target t and constant K, it is NP-complete to determine if there is a feasible solution $\mathbf{x}$ with expected cost $\mathbf{E}[C(\mathbf{W}^T\mathbf{x}-t)] \leq K$.*

Proof. The problem is in NP since there is a short certificate for it, given by the actual optimal solution if such exists.

To show NP-hardness, we reduce from the Subset Sum problem, namely given a set of integers $\{w_1, ..., w_n\}$ and a target integer t, is there a subset which sums exactly to t? The subset sum problem, which is a special case of the knapsack problem, is NP-complete [22]. Set $K = C(t_{min})$. Consider the graph in Figure 2.3 with deterministic edge travel times $w_1, ..., w_n$ on the bottom and 0 on the top. Any path P from S to T in this graph has travel time $\sum_{i\in P} w_i$ and cost $C(t' + \sum_{i\in P} w_i)$ if we leave the source at time $t' = t_{min} - t$. Since the cost function $C(t)$ has a unique global minimum at t_{min}, there is a path of cost at most $K = C(t_{min})$ if and only if there is a path with travel time satisfying $t' + \sum_{i\in P} w_i = t_{min}$, *i.e.,* if and only if there is a subset of the w_i's summing exactly to t. □

Remark 2.3.8. *Note that Theorem 2.3.1 only shows that it is NP-hard to find a simple optimal path. Theorem 2.3.7 on the other hand applies to non-simple paths as well since the subset sum problem is NP-complete even if it allows for repetitions [22].*

Algorithms for the optimal path with fixed start time

We have shown that finding simple optimal paths with a given departure time is hard to approximate within a constant factor, as it is similar to finding the longest path. When we remove the restriction to simple paths, we can give a pseudopolynomial dynamic programming algorithm, which depends polynomially on the largest mean travel time of a path M (or equivalently on the maximum expected travel time of a link). For real world applications such as car navigation, M will likely be polynomial in the size of the network, hence this algorithm would be efficient in practice, and significantly better than the previous algorithms based on exhaustive search [94].

For clarity, we present in detail algorithms for the quadratic and quadratic+exponential cost function families under general travel time distributions, for which we derived simple closed form expressions; however, the algorithms can readily extend to general polynomials and polynomials+exponentials.

Quadratic costs First consider the case of a quadratic cost where the expected cost of a path P with incidence vector $\mathbf{x}$ is $EC(\mathbf{x}, t) = (t + \boldsymbol{\mu}^T\mathbf{x})^2 + \boldsymbol{\tau}^T\mathbf{x}$. We are interested in finding the path of smallest cost $EC(\mathbf{x}, t)$, for a fixed t. Denote by $\pi(v, m)$ the predecessor of node v on the path from the source S to v of mean m and smallest variance. Denote by $\Phi(v, m)$ the variance of this path. Then we can find $\pi(v, m)$ and $\Phi(v, m)$ for all v and all $m = 0, 1, ..., M$ by considering the neighbors v' of v (denoted $v' \sim v$) and choosing the predecessor leading to smallest variance of the path from s to v:

$$\Phi(v, m) = \min_{v' \sim v} \left[\Phi(v', m - \mu_{v'v}) + \tau_{v'v}\right],$$

$$\pi(v, m) = arg\min_{v' \sim v} \left[\Phi(v', m - \mu_{v'v}) + \tau_{v'v}\right],$$

```
// Initialize paths out of source $S$ with mean 1
path variance $\Phi(S,0) := 0$; predecessor node $\pi(S,0) := S$
for each vertex $v$
    if $v$ is a neighbor of $S$ and $\mu_{sv} = 1$
        $\Phi(v,0) := \tau_{sv}$; $\pi(v,0) := S$
    else
        $\Phi(v,0)$ :=null ; $\pi(v,0)$ :=null

// Fill in the rest of the table
for $m$ = 1 to $M$
    for each vertex $v$ with neighbors $v'$
        $\Phi(v,m) := \min_{v' \sim v} \left[\Phi(v', m - \mu_{v'v}) + \tau_{v'v}\right]$
        $\pi(v,m) := arg\min_{v' \sim v} \left[\Phi(v', m - \mu_{v'v}) + \tau_{v'v}\right]$

// Find the lowest cost path from $S$ to $T$ at departure time $t$
$m_{opt} = \text{argmin}_{m \in \{0,\dots,M\}} \{(t+m)^2 + \Phi(T,m)\}$
$\pi_{opt} = \pi(T, m_{opt})$
$EC_{min}(t) = (t + m_{opt})^2 + \Phi(T, m_{opt})$
```

Figure 2.4. Pseudopolynomial Algorithm for Quadratic cost and fixed departure time. M is the upper bound on the mean travel time of a path.

where $\mu_{v'v}$ and $\tau_{v'v}$ denote the mean and variance of the travel time on the link (v', v). Note that, by our assumption of independence of the travel times, the variance of each path is given by the sum of variances of the edges. Hence suboptimality holds—the path of smallest variance from S to v through v' must be using a path of smallest variance from S to v'.

Suppose the maximum degree in the graph is d. Then computing the paths of smallest variance above for each vertex and each possible expected travel time from 0 to M can be done in time $O(Mdn)$. Finally, we find the path from S to destination T by taking the minimum of $(t+m)^2 + \Phi(T,m)$ over all $m = 0, 1, ..., M$, so that the total running time of the algorithm is $O(Mdn)$. If, instead of integer, the travel time means are discrete with discrete step ϵ, the running time would be $O(Mdn/\epsilon)$ for small degrees d, or $O(Mn^2/\epsilon)$ in general. The algorithm is summarized in Figure 2.4.

Quadratic+Exponential costs We can solve the case of a quadratic+exponential penalty similarly—only this time our dynamic programming table would have an extra dimension for possible values of the variance of a path and the table entries would contain the path with smallest exponential term $\prod_{e \in P} \text{E}[e^{W_e}]$. Denote by $\pi(v, m, \tau)$ the predecessor of node v on the path from the source s to v of total mean travel time m, total variance τ and smallest value of $\prod_{e \in P} \text{E}[e^{W_e}]$. Further denote by $\Phi(v, m, \tau)$ the value of $\prod_{e \in P} \text{E}[e^{W_e}]$ on this path. Then as before, once we have computed $\Phi(v, \mu, \tau)$ and $\pi(v, \mu, \tau)$ for all nodes v, path means $\mu \leq m - 1$ and variances $\tau = 0, 1, ..., M$, we can compute $\Phi(v, m, \tau)$ and $\pi(v, m, \tau)$ for all v and $\tau = 0, 1, ..., M$ by setting

$$\Phi(v, m, \tau) = \min_{v' \sim v} \left[\Phi(v', m - \mu_{v'v}, \tau - \tau_{v'v}) * \mathbf{E}[e^{W_{v'v}}]\right],$$

$$\pi(v, m, \tau) = arg \min_{v' \sim v} \left[\Phi(v', m - \mu_{v'v}, \tau - \tau_{v'v}) * \mathbf{E}[e^{W_{v'v}}]\right],$$

where $W_{v'v}$ is the stochastic weight of link (v', v) and we assume that the variance of a path is upper bounded by its expectation, so it is at most M. Correctness follows as above by noting that the subpath-optimality property holds for $\prod_{e \in P} \mathbf{E}[e^{W_e}]$. Similarly, we find the path of lowest expected cost from the source to the destination T by taking the minimum over $m = 0, 1, ..., M$ and $\tau = 0, 1, ..., M$ of $(t+m)^2 + \tau + \lambda e^{-t}\Phi(T, m, \tau)$. The running time is now $O((M/\epsilon)^2 dn)$ for discrete travel times with discrete step ϵ.

The standard technique of scaling, which turns a pseudopolynomial algorithm into a fully polynomial approximation scheme such as in the knapsack problem [126], would work here if the ratio of the maximum mean of an edge to the cost of the optimal solution is polynomially bounded and would fail otherwise. If we do not have a bound on this ratio, we cannot achieve a polynomial approximation scheme, either. Note that the ratio can be arbitrarily large if the optimal path has arbitrarily small variance, say under a quadratic cost function. Even if we lift the cost function by a constant so as to avoid zero values as part of its definition, we may still have a constant optimum cost compared to large mean travel times of edges so we cannot eliminate the dependence of the running times above on the maximum path mean M.

General polynomial plus exponential costs

The above dynamic programming algorithms extend to the case when the expected cost is a general polynomial (plus exponential) with a constant number of terms. Since it is not clear how the various terms trade off, we would have to keep track of each term individually in a separate dimension of the dynamic programming table, and the running time would scale as M to the power of the number of terms. Scenarios which would fall in this category include general polynomial (plus exponential) cost functions and additive edge distributions such as Gaussian, Gamma with a fixed width parameter, etc. Under these distributions, the expected cost of path travel time Y would depend only on the distribution of Y as opposed to that of each individual link on the path, and would therefore have a constant number of terms. For example, when the cost $C(Y)$ is a polynomial of degree l, the expected cost $\mathbf{E}[C(Y)]$ is a linear combination of the random variable Y's first l moments, as noted by Loui [84], and in this case the dynamic programming algorithm would have running time proportional to M^l if each moment is bounded by M.

Experimental evaluation

We ran the pseudopolynomial algorithms on grid graphs with up to 1600 nodes for the quadratic objective and up to 100 nodes for the quadratic+exponential objective. The former graph instances can be viewed as the problem of navigating from one corner of Manhattan to another; the latter as finding a path around a city through a highway network. Run-times were typically a few seconds while memory turned out to be limiting factor: In the case of quadratic objective the dynamic programming table is two-dimensional, and in the quadratic+exponential objective the table is three-dimensional, *i.e.*, cubic in the size of the graph. Given the memory constraint, we set the largest edge mean to 10 on the graphs with 1600 nodes and to 4 on the graphs with 100 nodes. The edge means and variances were generated uniformly at random. The memory usage of the algorithms was not optimized; it could be made an order of magnitude smaller (linear for $C(t) = t^2$, quadratic for $C(t) = t^2 + \lambda e^{kt}$) if one only wanted to compute the objective function values without outputting the actual paths.

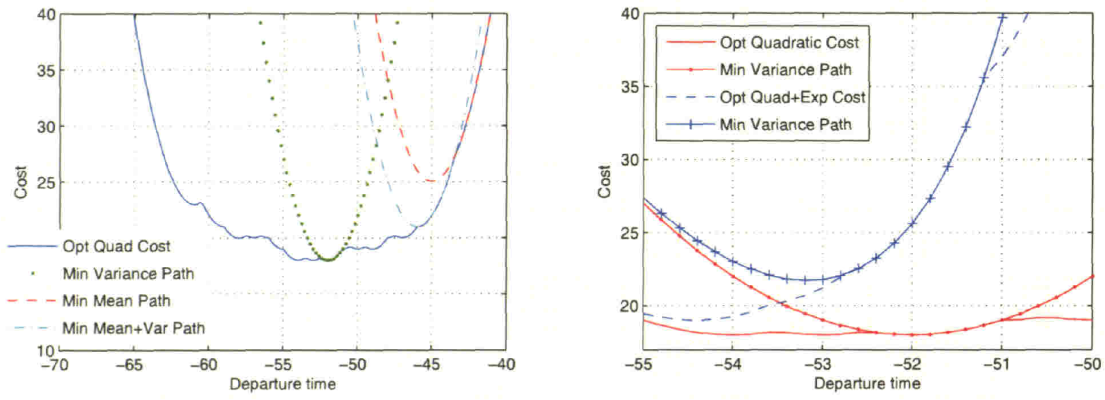


Figure 2.5. Minimum-cost envelopes for the same network under quadratic (top left) and quadratic+exponential (top right) cost functions. The envelopes are superimposed in the bottom graph in a neighborhood of their global optima.

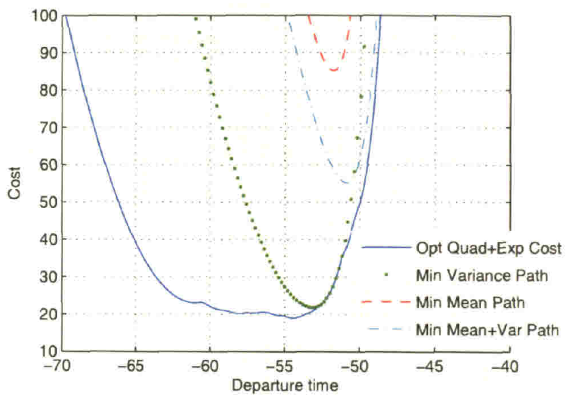
For the sake of comparison, Figure 2.5 shows the resulting optimal cost for the same graph with 100 nodes under both quadratic and quadratic+exponential objectives with Gamma distributed travel times. Recall that the optimal cost envelope for a graph under a given objective is the infimum of the cost functions of each path in the graph. Each plot on the top shows the minimum-cost envelope and the expected cost function of three paths—the path with the smallest variance, the smallest mean, and the smallest mean+variance. As predicted, the path with smallest variance yields the globally optimal cost in the quadratic case but this is not necessarily true in the quadratic+exponential case.

We note that the basic quadratic+exponential objective

$$\left(t+\sum_{i=1}^{r} a_i b_i\right)^2+\sum_{i=1}^{r} a_i b_i^2+e^t\left[\prod_{i=1}^{r}(1-b_i)^{-a_i}\right],$$

tends to be dominated by the quadratic term near the function's minimum so that its plot is almost identical to the plot of the quadratic objective case for the interval of departure times around the global minimum. However, a bigger positive coefficient in front of the exponential term (featured at the top right of Figure 2.5) balances the quadratic and exponential influence and illustrates a situation where the smallest variance path is not globally optimal and the smallest mean or mean+variance paths are not even locally optimal, *i.e.*, do not participate in the min-cost envelope.

The third plot in the Figure superimposes the two different objectives and zooms into their global minima. The plot demonstrates clearly the qualitative difference between the two objective



costs—not only in the fact that the global optimum is attained on different paths but also in that different paths may be locally optimal at the same fixed departure time. For example, at departure 51 minutes (units) before the deadline, a quadratic objective would recommend the smallest variance path while a quadratic+exponential objective would recommend some other path. We see similar differences of recommendation for departure time a little over 52.5 minutes before the deadline. Naturally, since the exponential term assigns a more severe penalty for being late, the quadratic+exponential objective recommends an earlier globally optimal departure time.

2.4 Monotonically increasing costs

The general stochastic problem (2.2) becomes significantly easier if we consider some natural monotonically increasing costs, such as linear and exponential for which the global cost is separable into element costs. As noted above, linear cost with a given threshold t translates to minimizing the expected solution weight, a basically uninteresting quantity in this stochastic setting. An exponential cost $C(t) = e^{kt}$ on the other hand, *is* interesting because it gives rise to the expected solution cost

$$e^{kt} \prod_{i:\, x_i=1} \mathbf{E}[e^{kW_i}].$$

where $\mathbf{x} = (x_1, ..., x_n)$ is the indicator vector of the solution and W_i is the stochastic weight of element i. We make this separable by moving into the log domain where finding the solution of lowest expected cost with given threshold t turns out to be equivalent to finding the minimum weight solution with deterministic weights set to the cumulant-generating function $K(k) = \log\left(\mathbf{E}[e^{kW_i}]\right)$. The cumulant-generating function is a series sum over cumulants that is dominated by the lowest central moments of the distribution; for many distributions it is effectively a weighted sum of mean and variance, a common objective in portfolio selection and mean-risk analysis [36]. For gamma-distributed weights, $K(k) = a\log(1/(1-kb))$; compare this with the more familiar expression $K(k) = k\mu + k^2\tau/2$ which arises from normally distributed variables.

CHAPTER 3

Nonconvex and stochastic optimization with risk

In the previous chapter, we defined a general framework for stochastic optimization that incorporates risk, based on a user-dependent utility or cost function $C(.)$, where the objective is to find the feasible solution x of smallest expected cost $\mathbf{E}\left[C(\mathbf{W}^T\mathrm{x})\right]$. We showed that for broad classes of cost functions C the problem is NP-hard and for arbitrary such costs it is essentially hopeless to solve or approximate. Therefore, in this chapter we focus on particular classes of costs which are both very natural and also admit efficient approximation schemes for arbitrary underlying problems.

The stochasticity here is defined through arbitrary independent distributions with given mean and variance values. We consider a family of objective functions specified by a convex combination of the mean and standard deviation of the feasible object. These functions and their closely related value-at-risk objectives, which we consider in a subsequent section, have been ubiquitously used in finance for portfolio optimization [36, 47].

This stochastic framework is a special case of nonconvex optimization, which is generally a very hard problem even over a hypercube [50], let alone over a complex combinatorial or discrete structure whose convex hull cannot be efficiently described.

Our main contribution in this chapter is a family of general-purpose approximation schemes for minimizing the objective $\boldsymbol{\mu}^T\mathrm{x} + k\sqrt{\boldsymbol{\tau}^T\mathrm{x}}$ *for an arbitrary underlying problem (namely an arbitrary feasible set* $\mathcal{F}$*), where* $\mathrm{x}, \boldsymbol{\mu}, \boldsymbol{\tau}$ *are the vectors corresponding to a feasible solution, the vector of means and the vector of variances respectively. Our algorithms assume oracle access to an* exact *or* approximate *algorithm for solving the deterministic (linear) version of the problem* $\{\min \mathrm{w}^T\mathrm{x} \mid \mathrm{x} \in \mathcal{F}\}$. *We show how our approximation algorithms technique extends to other objective functions involving the probability tail of the stochastic solution weight such as the so called value-at-risk objective and a threshold objective defined later in the chapter.*

The main insight is that as an instance of concave minimization, *the optimum lies in the union of hyperplanes (in contrast to convex minimization where the optimum is in the intersection of hyperplanes). Although we may need exponentially or infinitely many hyperplanes to find the exact optimum, a careful choice of a* small *number of them yields a very good approximation in polynomial time. That choice is related to the field of polytopal approximation of convex bodies, where the goal is to approximate a convex body by a polytope with the smallest number of facets.*

Our work advances the emerging algorithmic theory of nonlinear discrete optimization, where the state of art consists of algorithms that enumerate feasible extreme points or discretize the state space [102, 78] and do not apply to settings such as ours. On the other hand, our stochastic optimization framework with risk constitutes an important contribution in stochastic optimization in its own right, offering a unified approximative analysis that applies to arbitrary combinatorial or continuous problems.

We defer further discussion of our contribution and related work after formally describing the relevant problems and results.

3.1 Approximation technique for nonconvex optimization

Our results in this chapter are approximation algorithms for the problem of minimizing $\mu^T\mathbf{x} + k\sqrt{\tau^T\mathbf{x}}$ *and two other related objectives over arbitrary feasible sets. Since we use the same approximation technique, which we conjecture will also apply to other objective functions of interest, in this section we extract and give a nontechnical description of our recipe for obtaining these algorithms.*

Consider an arbitrary feasible set $\mathcal{F} \subset \mathbb{R}^n$ (in particular, the set can have a combinatorial or more generally discrete or continuous, possibly nonconvex, structure). We call points $\mathbf{x} \in \mathcal{F}$ from the set feasible objects or feasible solutions. Now let $f : \mathbb{R}^n \to \mathbb{R}$ be a function which we wish to optimize (say minimize) over the feasible set:

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \mathcal{F}. \end{array} \tag{3.1}$$

Stated in its full generality as in Equation (3.1), this problem contains as special cases linear programming, integer programming, combinatorial optimization, concave minimization, etc. and an attempt at a general approach to solve it would likely fail. It does not get much easier if we assume access to an oracle which can optimize linear functions over the feasible set:

$$\begin{array}{ll} \text{minimize} & \mathbf{w}^T\mathbf{x} \\ \text{subject to} & \mathbf{x} \in \mathcal{F}, \end{array} \tag{3.2}$$

where we refer to $\mathbf{w} \in \mathbb{R}^n$ as a weight vector or linear objective. In this case, even just quadratic concave programming over the hypercube is NP-hard [50], so it is not of much help that we can optimize linear functions over the hypercube. Nevertheless, this is the framework that we adopt: assuming access to an algorithm for optimizing linear objectives (3.2), which may be *exact* or *approximate* (we refer to it as the *linear oracle*), we would like to approximate the nonlinear problem (3.1), by calling the linear oracle on a carefully chosen set of linear objectives. As we demonstrate later, such an approach enables solving large classes of problems all at once by providing a family of corresponding general-purpose algorithms.

Ours is by no means the first work to take the direction of reducing nonlinear (or linear integer) to linear problems. The idea is present in classical work on optimization; the use of cutting plane techniques is one of many examples. The subtlety of course lies in precisely how to choose appropriate linear objectives given the problem at hand, so as to ensure a fast and effective approximation.

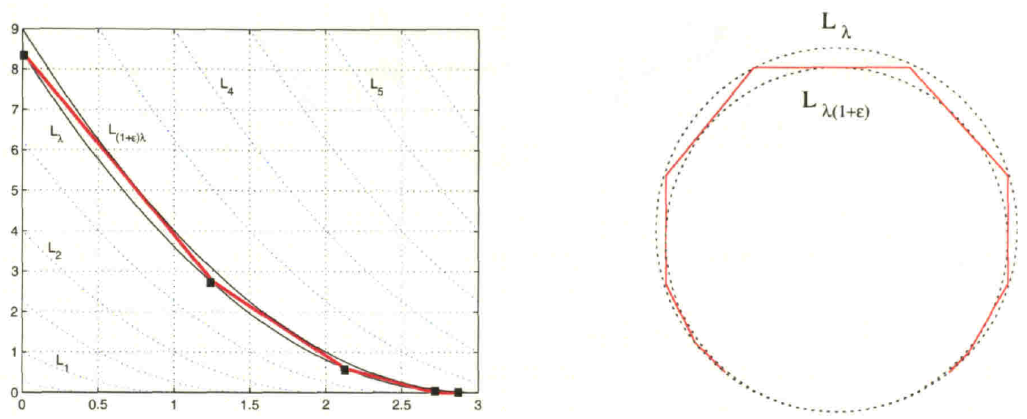


Figure 3.1. *(left)* Level sets and approximate nonlinear separation oracle for the projected nonconvex (stochastic) objective $f(\mathbf{x}) = \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}}$ on the $span(\boldsymbol{\mu}, \boldsymbol{\tau})$-plane. *(right)* Approximating the level sets L_λ of an objective function with a large or unbounded gradient as our threshold objective may require a large or even infinite number of linear segments.

The methodology we propose gives a recipe for how to choose a set of linear objectives for approximating a given nonconvex program. In this work, we show how it successfully applies to an entire area of stochastic optimization problems with risk. Beyond the applications discussed here, we conjecture that the methodology would yield practical polynomial-time approximation algorithms for (quasi)concave minimization,[1] (quasi)convex maximization and other nonconvex functions of low rank (*i.e.,* that can be thought of as functions on a low-dimensional subspace), attaining their maxima at extreme points of the feasible set.

■ 3.1.1 Building an approximate nonlinear oracle from linear oracle

The key idea is to approximate the nonlinear *level sets* of the objective function by as few linear segments as possible (see figure 3.1). Denote the level set for function value λ by $L_\lambda = \{\mathbf{x} \mid f(\mathbf{x}) = \lambda\}$. We would sometimes use the *lower* and *upper level sets* for value λ, on which the function takes values $\leq \lambda$ and $\geq \lambda$ respectively. Formally we denote them as $\underline{L}_\lambda = \{\mathbf{x} \mid f(\mathbf{x}) \leq \lambda\}$ and $\bar{L}_\lambda\{\mathbf{x} \mid f(\mathbf{x}) \geq \lambda\}$.

If we want to achieve a $(1+\epsilon)$-approximation for some $\epsilon > 0$, we consider the level sets L_λ and $L_{\lambda(1+\epsilon)}$ and construct a sequence of hyperplanes tangent to one of the two level sets, which form a (not necessarily convex) polyhedron whose facets fit entirely between the two level sets. For example, in dimension 2, we draw tangents to the level sets as shown in figure 3.1, which result in a polygon whose sides are entirely contained between the two level sets. It is not the focus of this work to prove that this yields the smallest number of linear objectives that are necessary to approximate the level set L_λ, though this seems intuitive from the construction and it would be nice to settle formally in the future. However, it is our goal to show that the resulting number of linear objectives in this construction is small for problems of interest and gives rise to an efficient approximation of the level sets.

[1]See Appendix for definitions of quasiconcave functions.

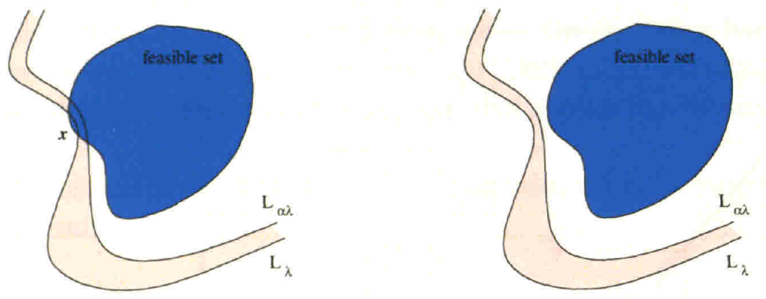


Figure 3.2. An α-approximate nonlinear separation oracle either returns a feasible solution $\mathbf{x} \in \mathcal{F}$ satisfying $f(\mathbf{x}) \leq \alpha\lambda$ or asserts that $f(\mathbf{x}) > \lambda$ for all $\mathbf{x} \in \mathcal{F}$, that is the entire feasible set is on one side of the level set L_λ.

Now, we formally define the approximate nonlinear oracle and show how when one is available, we can solve the nonconvex program (3.1).

Definition 3.1.1 (Approximate Nonlinear Separation Oracle). *An α-approximate nonlinear oracle ($\alpha \geq 1$) for minimizing an objective function $f : \mathbb{R}^n \to \mathbb{R}$ over a feasible set $\mathcal{F} \subset \mathbb{R}^n$ solves the following nonlinear separation problem: given a function value λ, the oracle returns either*

1. *A feasible point $\mathbf{x} \in \mathcal{F}$ with $f(\mathbf{x}) \leq \alpha\lambda$,* *or*
2. *An answer that $f(\mathbf{x}) > \lambda$ for all $\mathbf{x} \in \mathcal{F}$.*

Of course one can analogously define the corresponding oracle for a maximization problem.

3.1.2 Using the nonlinear oracle to approximate the function optimum

Once we have a nonlinear separation oracle, it is straightforward to approximate the function optimum via a search on the possible values.

Lemma 3.1.2. *Given an α-nonlinear separation oracle ($\alpha > 1$) for the problem of minimizing a function $f : \mathbb{R}^n \to \mathbb{R}$ over a feasible set $\mathcal{F} \subset \mathbb{R}$, along with a lower bound $f_l > 0$*[2] *and upper bound f_u on the function minimum, there is an $\alpha(1+\xi)$-approximate algorithm which finds the minimum via at most $\log(\frac{f_u}{f_l})/\log(1+\xi)$ calls to the nonlinear oracle, for any desired accuracy $\xi > 0$.*

Proof. Apply the nonlinear oracle successively on the sequence of function values f_l, $(1+\xi)f_l$, $(1+\xi)^2 f_l$, ..., until we reach a level $\lambda = (1+\xi)^i f_l \leq f_u$ for which the oracle returns a feasible solution $\mathbf{x}'$ with

$$f(\mathbf{x}') \leq \alpha\lambda = \alpha(1+\xi)^i f_l.$$

From running the oracle on the previous level $(1+\xi)^{i-1} f_l$, we know that $f(\mathbf{x}) \geq f(\mathbf{x}_{opt}) > (1+\xi)^{i-1} f_l$ for all $\mathbf{x}$ in the feasible set, where $\mathbf{x}_{opt}$ denotes the optimal solution. Therefore,

$$(1+\xi)^{i-1} f_l < f(\mathbf{x}_{opt}) \leq f(\mathbf{x}') \leq \alpha(1+\xi)^i f_l, \qquad \text{and hence}$$

$$f(\mathbf{x}_{opt}) \leq f(\mathbf{x}') < \alpha(1+\xi) f(\mathbf{x}_{opt}).$$

[2]We allow $f_l \geq 0$ for combinatorial problems, where the case $f_l = 0$ is handled separately.

So the solution $\mathbf{x}'$ gives an $\alpha(1+\xi)$-approximation to the optimum $\mathbf{x}_{opt}$. In the process, we run the approximate nonlinear separation oracle at most $\log(\frac{f_u}{f_l})/\log 1+\xi$ times, QED. □

3.1.3 Complexity of the nonlinear separation oracle

So far we have explained the easy conceptual part of how to construct a nonlinear oracle for the level sets of the objective function from appropriately selected linear pieces, and how, given this oracle, we can approximate the optimal solution.

The main question remains of how many linear oracle queries we need in order to construct the desired nonlinear oracle. Even in two dimensions, we can see from figure 3.1 that depending on the mutual position and curvature of the function level sets, we may need a large or even infinite number of linear segments. Another intuitive description of such situation is that when the objective function has unbounded or infinite gradient, no matter how close two points $\mathbf{x}_1$ and $\mathbf{x}_2$ are in this region, their corresponding objective values $f(\mathbf{x}_1)$ and $f(\mathbf{x}_2)$ can still be far apart.

One of the objective function families we consider in the later sections has an unbounded gradient and as an additional challenge it can attain value 0 (as opposed to strictly positive). In light of the above discussion it is not clear that approximating the level sets and the function minimum up to a multiplicative factor is even possible, let alone in a way that uses a finite or polynomial number of linear oracle calls. It is perhaps a counterintuitive positive result that we can achieve an arbitrarily good approximation with just a logarithmic number of linear objectives.

In the subsequent sections the crux of the technical work is in establishing logarithmic complexity of the nonlinear oracle for families of objective functions arising in stochastic optimization with risk, as well as extending the construction to the case when we have *approximate* instead of *exact* linear oracles.

Related Work On the nonconvex optimization side, our work advances the emerging (approximation) algorithmic theory of nonlinear discrete optimization, as referred to by Onn and collaborators (see survey [102] and references therein). Like us, Onn *et al.* solve nonlinear programs via calls to (only exact) linear oracles. Albeit admitting more general objective function specifications they impose very restrictive conditions on the feasible sets. These conditions ensure that there are a polynomial or pseudo-polynomial number of relevant extreme points (and exclude basic combinatorial problems like shortest paths). From there on, they give exact (pseudo-)polynomial algorithms which exhaustively enumerate extreme points. The authors also provide constant-approximation pseudo-polynomial algorithms for combinatorial norm maximization [9]. Naturally, the approach of enumerating extreme points does not extend to continuous feasible sets with infinitely many extreme points, as ours does.

Kelner and Nikolova [78] provide expected polynomial-time approximation algorithms for minimizing low-rank quasiconcave functions with bounded gradients and strictly positive (as opposed to nonnegative) bound on the minimum, also based on enumerating relevant extreme points. These algorithms are described in Chapter 4. In addition to using entirely different techniques, our results in this Chapter are different in that they extend to non-polyhedral feasible sets and unbounded gradients. Independently from Kelner and Nikolova, Goyal also gives approximation algorithms for low-rank quasiconcave minimization by invoking exact linear oracles over a discretization of the state space [56].

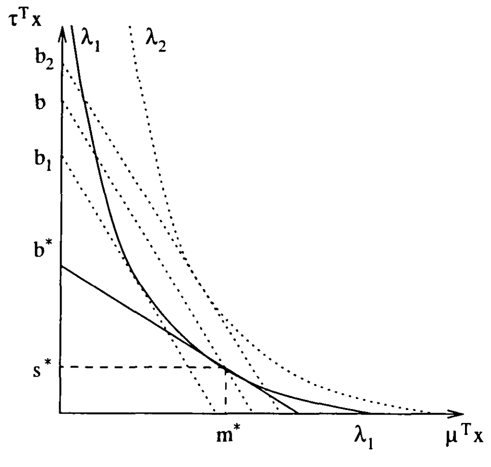


Figure 3.3. Approximating the objective value λ_1 of the optimal solution (m^*, s^*).

On the stochastic optimization side, our results are similar in spirit to work on robust optimization [12], in which robust combinatorial problems are solved via oracle calls to the corresponding deterministic (linear) problems (though beyond this very high-level similarity, our models, results and techniques are entirely different). The probability tail objective from Sec. 3.4.1 was previously considered in the context of stochastic shortest paths [99] and an exact algorithm was given based on enumerating relevant extreme points from the path polytope.

A wealth of different models for stochastic combinatorial optimization have appeared in the literature, perhaps most commonly on two-stage and multi-stage stochastic optimization, *e.g.,* [119, 62, 77, 61, 68], see survey by Swamy and Shmoys [123]. Almost all such work considers linear objective functions (*i.e.,* minimizing the expected solution weight) and as such does not consider risk. Some of the models incorporate additional budget constraints [121] or threshold constraints for specific problems such as knapsack, load balancing and others [29, 52, 80]. A survey on stochastic optimization with risk with a different focus (continuous optimization settings and different solution concepts) is provided by Rockafellar [115]. Similarly, continuous optimization work with probability (chance) constraints (*e.g.,* [97]) applies for linear and *not* discrete optimization problems. Additional related work on the combinatorial optimization side includes research on multi-criteria optimization, *e.g.,* [104, 2, 117, 128] and combinatorial optimization with a ratio of linear objectives [89, 113].

■ 3.2 Approximating stochastic optimization with risk

In this section we present a framework for stochastic optimization with risk and give for it a family of general-purpose approximation algorithms based on the nonconvex optimization technique described in the previous section.

Again suppose we have an arbitrary set of feasible objects $\mathcal{F} \subset \mathbb{R}^n$, which may be discrete or continuous, together with an oracle for optimizing linear objectives over the set. In addition we are given a vector of means $\mu \in \mathbb{R}^n$ and variances $\tau \in \mathbb{R}^n$ for the stochastic weights, coming from arbitrary independent distributions so that the mean and variance of an object $\mathrm{x} \in \mathbb{R}^n$ is $\mu^T \mathrm{x}$

and $\boldsymbol{\tau}^T\mathbf{x} \geq 0$ respectively. We are interested in finding a feasible solution which has optimal cost, where the notion of optimality incorporates risk. A family of objectives that have been widely used in finance is the class of convex combinations of mean and standard deviation [36, 47]. We adopt this model here. In later sections, we look at other models which incorporate the probability distribution tails in the objective or constraint. Formally, our problem is to:

$$\begin{aligned} &\text{minimize} && \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}} \\ &\text{subject to} && \mathbf{x} \in \mathcal{F}, \end{aligned} \tag{3.3}$$

where the constant c parametrizes the degree of the user's risk aversion. The objective function is a concave function on $\mathbb{R}^n$; however, thanks to its form, it can be projected onto $span(\boldsymbol{\mu}, \boldsymbol{\tau})$ and can be thought of as a function on two dimensions. In the plane $span(\boldsymbol{\mu}, \boldsymbol{\tau})$, the projected level sets of the objective function $f(\mathbf{x}) = \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}}$ are parabolas. We follow the idea from Section 3.1.1 on which linear segments to approximate the level sets with. Geometrically, the optimal choice of segments starts from one endpoint of the level set L_λ and repeatedly draws tangents to the level set $L_{(1+\epsilon)\lambda}$, as shown in Figure 3.3.

In order to establish that the resulting linear segments are few, we first show that the tangent-segments to $L_{(1+\epsilon)\lambda}$ starting at the endpoints of L_λ are sufficiently long.

Lemma 3.2.1. *Consider points (m_1, s_1) and (m_2, s_2) on L_λ with $0 \leq m_1 < m_2 \leq \lambda$ such that the segment with these endpoints is tangent to $L_{(1+\epsilon)\lambda}$ at point $\alpha(m_1, s_1) + (1-\alpha)(m_2, s_2)$. Then $\alpha = \frac{c^2}{4}\frac{s_1-s_2}{(m_2-m_1)^2} - \frac{s_2}{s_1-s_2}$ and the objective value at the tangent point is $\left[\frac{c^2}{4}\frac{s_1-s_2}{m_2-m_1} + s_2\frac{m_2-m_1}{s_1-s_2} + m_2\right]$.*

Proof. Let $\bar{f} : \mathbb{R}^2 \to \mathbb{R}$, $\bar{f}(m, s) = m + c\sqrt{s}$ be the projection of the objective $f(\mathbf{x}) = \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}}$. The objective values along the segment with endpoints (m_1, s_1), (m_2, s_2) are given by

$$h(\alpha) = \bar{f}(\alpha m_1 + (1-\alpha)m_2, \alpha s_1 + (1-\alpha)s_2) = \alpha(m_1 - m_2) + m_2 + c\sqrt{\alpha(s_1 - s_2) + s_2},$$

for $\alpha \in [0, 1]$. The point along the segment with maximum objective value (that is, the tangent point to the minimum level set bounding the segment) is found by setting the derivative $h'(\alpha) = m_1 - m_2 + c\frac{s_1-s_2}{2\sqrt{\alpha(s_1-s_2)+s_2}}$, to zero:

$$\begin{aligned} m_2 - m_1 &= c\frac{s_1 - s_2}{2\sqrt{\alpha(s_1 - s_2) + s_2}} \\ \Leftrightarrow \quad \sqrt{\alpha(s_1 - s_2) + s_2} &= c\frac{s_1 - s_2}{2(m_2 - m_1)} \\ \Leftrightarrow \quad \alpha(s_1 - s_2) + s_2 &= c^2\frac{(s_1 - s_2)^2}{4(m_2 - m_1)^2} \\ \Leftrightarrow \quad \alpha(s_1 - s_2) &= c^2\frac{(s_1 - s_2)^2}{4(m_2 - m_1)^2} - s_2 \\ \Leftrightarrow \quad \alpha &= c^2\frac{s_1 - s_2}{4(m_2 - m_1)^2} - \frac{s_2}{s_1 - s_2}. \end{aligned}$$

This is a maximum, since the derivative $h'(\alpha)$ is decreasing in α. The objective value at the maximum is

$$\begin{aligned} h(\alpha_{\max}) &= \alpha_{\max}(m_1 - m_2) + m_2 + c\sqrt{\alpha_{\max}(s_1 - s_2) + s_2} \\ &= \left[c^2 \frac{s_1 - s_2}{4(m_2 - m_1)^2} - \frac{s_2}{s_1 - s_2}\right](m_1 - m_2) + m_2 + c^2 \frac{s_1 - s_2}{2(m_2 - m_1)} \\ &= -\frac{c^2}{4}\frac{s_1 - s_2}{m_2 - m_1} - s_2 \frac{m_1 - m_2}{s_1 - s_2} + m_2 + \frac{c^2}{2}\frac{s_1 - s_2}{m_2 - m_1} \\ &= \frac{c^2}{4}\frac{s_1 - s_2}{m_2 - m_1} + s_2 \frac{m_2 - m_1}{s_1 - s_2} + m_2. \end{aligned}$$

Further, since $s_1 = (\frac{\lambda - m_1}{c})^2$ and $s_2 = (\frac{\lambda - m_2}{c})^2$, their difference satisfies $s_1 - s_2 = \frac{1}{c^2}(m_2 - m_1)(2\lambda - m_1 - m_2)$, so $\frac{s_1 - s_2}{m_2 - m_1} = \frac{2\lambda - m_1 - m_2}{c^2}$ and the above expression for the maximum function value on the segment becomes

$$h(\alpha_{\max}) = \frac{c^2}{4}\frac{2\lambda - m_1 - m_2}{c^2} + \frac{c^2 s_2}{2\lambda - m_1 - m_2} + m_2 = \frac{2\lambda - m_1 - m_2}{4} + \frac{(\lambda - m_2)^2}{2\lambda - m_1 - m_2} + m_2.$$

□

Lemma 3.2.2. *Consider the endpoint* $(m_2, s_2) = (\lambda, 0)$ *of* L_λ. *Then either the single segment connecting the two endpoints of* L_λ *is entirely below the level set* $L_{(1+\epsilon)\lambda}$, *or the other endpoint of the segment tangent to* $L_{(1+\epsilon)\lambda}$ *is* $(m_1, s_1) = (\lambda(1 - 4\epsilon), (\frac{4\epsilon\lambda}{c})^2)$.

Proof. Since $0 \le m_1 < \lambda$, we can write $m_1 = \beta\lambda$ for some $\beta \in [0, 1)$. Consequently, $s_1 = (\frac{\lambda - m_1}{c})^2 = \frac{\lambda^2(1-\beta)^2}{c^2}$ and $\frac{s_1 - s_2}{m_2 - m_1} = \frac{\lambda^2(1-\beta)^2}{c^2\lambda(1-\beta)} = \frac{\lambda(1-\beta)}{c^2}$. By Lemma 3.2.1, the objective value at the tangent point is

$$\frac{c^2}{4}\frac{\lambda(1-\beta)}{c^2} + \lambda = \lambda\left(\frac{1-\beta}{4} + 1\right) = (1+\epsilon)\lambda.$$

The last equality follows by our assumption that the tangent point lies on the $L_{(1+\epsilon)\lambda}$ level set. Hence, $\beta = 1 - 4\epsilon$, so $m_1 = (1 - 4\epsilon)\lambda$ and $s_1 = (\frac{\lambda - m_1}{c})^2 = (\frac{4\epsilon\lambda}{c})^2$. □

Next, we characterize the segments with endpoints on L_λ that are tangent to the level set $L_{\lambda(1+\epsilon)}$.

Lemma 3.2.3. *Consider two points* (m_1, s_1), (m_2, s_2) *on* L_λ *with* $0 \le m_1 < m_2 \le \lambda$ *and such that the segment connecting the two points is tangent to* $L_{(1+\epsilon)\lambda}$. *Then the ratio* $\frac{s_1}{s_2} \ge (1 + 2\epsilon)^2$.

Proof Continuation. We showed that $\frac{s_1}{s_2}$ satisfies the following quadratic equation:

$$\frac{s_1}{s_2} + 1 - \frac{4\epsilon\lambda}{c\sqrt{s_2}}(\sqrt{\frac{s_1}{s_2}} + 1) - 2\sqrt{\frac{s_1}{s_2}} = 0$$

Denote for simplicity $z = \sqrt{\frac{s_1}{s_2}}$ and $w = \frac{2\epsilon\lambda}{c\sqrt{s_2}}$, then we have to solve the following quadratic equation for z in terms of w:

$$\begin{aligned} & z^2 + 1 - 2w(z+1) - 2z = 0 \\ \Leftrightarrow\ & z^2 - 2z(w+1) + 1 - 2w = 0. \end{aligned}$$

The discriminant of this quadratic expression is $D = (w+1)^2 - 1 + 2w = w^2 + 4w$ and its roots are $z_{1,2} = 1 + w \pm \sqrt{w^2 + 4w}$, Since $\frac{s_1}{s_2} > 1$, we choose the bigger root $z_2 = 1 + w + \sqrt{w^2 + 4w}$. Therefore since $w = \frac{2\epsilon\lambda}{c\sqrt{s_2}} \geq 0$ we have

$$\sqrt{\frac{s_1}{s_2}} = 1 + w + \sqrt{w^2 + 4w} \geq 1 + w = 1 + \frac{2\epsilon\lambda}{c\sqrt{s_2}} \geq 1 + \frac{2\epsilon\lambda}{c\frac{\lambda}{c}} = 1 + 2\epsilon,$$

where the last inequality follows from the fact that $\sqrt{s_2} < \sqrt{s_1} \leq \frac{\lambda}{c}$. This concludes the proof. □

Proof. Let point (m, s) on the segment with endpoints (m_1, s_1), (m_2, m_2) be the tangent point to the level set $L_{(1+\epsilon)\lambda}$. Then the slope $\frac{s_1 - s_2}{m_1 - m_2}$ of the segment is equal to the derivative of the function $y = (\frac{(1+\epsilon)\lambda - x}{c})^2$ at $x = m$, which is $-2\frac{(1+\epsilon)\lambda - m}{c^2} = -\frac{2\sqrt{s}}{c}$. Since $\frac{s_1 - s_2}{m_1 - m_2} = \frac{s_1 - s_2}{(\lambda - m_2) - (\lambda - m_1)} = \frac{s_1 - s_2}{c(\sqrt{s_2} - \sqrt{s_1})} = -\frac{\sqrt{s_2} + \sqrt{s_1}}{c}$, equating the two expressions for the slope we get $2\sqrt{s} = \sqrt{s_2} + \sqrt{s_1}$.

On the other hand, since $(m, s) \in L_{(1+\epsilon)\lambda}$, we have

$$m = (1+\epsilon)\lambda - c\sqrt{s} = (1+\epsilon)\lambda - \frac{c\sqrt{s_2} + c\sqrt{s_1}}{2} = (1+\epsilon)\lambda - \frac{\lambda - m_2 + \lambda - m_1}{2} = \epsilon\lambda + \frac{m_1 + m_2}{2}$$

and $m = \alpha(m_1 - m_2) + m_2$ for some $\alpha \in (0, 1)$. Therefore $\alpha = \frac{1}{2} - \frac{\epsilon\lambda}{m_2 - m_1} = \frac{1}{2} - \frac{\epsilon\lambda}{c(\sqrt{s_1} - \sqrt{s_2})}$.

Next,

$$\begin{aligned} s &= \alpha(s_1 - s_2) + s_2 = \left[\frac{1}{2} - \frac{\epsilon\lambda}{c(\sqrt{s_1} - \sqrt{s_2})}\right](s_1 - s_2) + s_2 = \frac{s_1 - s_2}{2} - \frac{\epsilon\lambda}{c}(\sqrt{s_1} + \sqrt{s_2}) + s_2 \\ &= \frac{s_1 + s_2}{2} - \frac{\epsilon\lambda}{c}(\sqrt{s_1} + \sqrt{s_2}) \end{aligned}$$

therefore using $2\sqrt{s} = \sqrt{s_2} + \sqrt{s_1}$ from above, we get two equivalent expressions for $4s$:

$$\begin{aligned} & 2(s_1 + s_2) - \frac{4\epsilon\lambda}{c}(\sqrt{s_1} + \sqrt{s_2}) = s_1 + s_2 + 2\sqrt{s_1 s_2} \\ \Leftrightarrow\ & s_1 + s_2 - \frac{4\epsilon\lambda}{c}(\sqrt{s_1} + \sqrt{s_2}) - 2\sqrt{s_1 s_2} = 0 \\ \Leftrightarrow\ & \frac{s_1}{s_2} + 1 - \frac{4\epsilon\lambda}{c\sqrt{s_2}}(\sqrt{\frac{s_1}{s_2}} + 1) - 2\sqrt{\frac{s_1}{s_2}} = 0 \end{aligned}$$

Denote for simplicity $z = \sqrt{\frac{s_1}{s_2}}$ and $w = \frac{2\epsilon\lambda}{c\sqrt{s_2}}$, then we have to solve the following quadratic equation for z in terms of w:

$$\begin{aligned} & z^2 + 1 - 2w(z+1) - 2z = 0 \\ \Leftrightarrow\ & z^2 - 2z(w+1) + 1 - 2w = 0. \end{aligned}$$

The discriminant of this quadratic expression is $D = (w+1)^2 - 1 + 2w = w^2 + 4w$ and its roots are $z_{1,2} = 1 + w \pm \sqrt{w^2+4w}$, Since $\frac{s_1}{s_2} > 1$, we choose the bigger root $z_2 = 1 + w + \sqrt{w^2+4w}$. Therefore since $w = \frac{2\epsilon\lambda}{c\sqrt{s_2}} \geq 0$ we have

$$\sqrt{\frac{s_1}{s_2}} = 1 + w + \sqrt{w^2+4w} \geq 1 + w = 1 + \frac{2\epsilon\lambda}{c\sqrt{s_2}} \geq 1 + \frac{2\epsilon\lambda}{c\frac{\lambda}{c}} = 1 + 2\epsilon,$$

where the last inequality follows from the fact that $\sqrt{s_2} < \sqrt{s_1} \leq \frac{\lambda}{c}$. This concludes the proof. □

The previous lemma shows that each segment is sufficiently long so that overall the number of tangent segments approximating the level set L_λ is small, in particular, it is polynomial in $\frac{1}{\epsilon}$ (and does not depend on the input size of the problem!). This gives us the desired approximate nonlinear separation oracle for the level sets of the objective function.

Lemma 3.2.4. *A nonlinear* $(1+\epsilon)$*-approximate separation oracle to any level set of the nonconvex objective* $f(\mathrm{x})$ *in problem (3.3) can be found with* $(1 + \frac{\log(\frac{1}{16\epsilon^2})}{2\log(1+2\epsilon)})$ *queries to a linear oracle for solving the underlying linear (deterministic) problem (3.2).*

The nonlinear oracle takes as inputs λ, ϵ *and returns either a feasible solution* $\mathrm{x} \in \mathcal{F}$ *with* $f(\mathrm{x}) \leq (1+\epsilon)\lambda$ *or an answer that* $f(\mathrm{x}) > \lambda$ *for all* x *in polytope*$(\mathcal{F})$.

Proof. Apply the available linear oracle to the slopes of the segments with endpoints on the specified level set, say L_λ, and which are tangent to the level set $L_{(1+\epsilon)\lambda}$. By Lemma 3.2.3 and Lemma 3.2.2, the y-coordinates of endpoints of these segments are given by $s_1 = (\frac{\lambda}{c})^2$, $s_2 \leq \frac{s_1}{(1+2\epsilon)^2}$, $s_3 \leq \frac{s_1}{(1+2\epsilon)^4}, \dots$, $s_k \leq \frac{s_1}{(1+2\epsilon)^{2(k-1)}}$, $s_{k+1} = 0$, where $s_k = (\frac{4\epsilon\lambda}{c})^2$, so $k = 1 + \log(\frac{1}{16\epsilon^2})/2\log(1+2\epsilon)$, which is precisely the number of segments we use and the result follows. □

Finally, combining the approximate non-linear separation oracle from Lemma 3.2.4 and Lemma 3.1.2 gives an approximation algorithm for the nonconvex problem (3.3). In a combinatorial problem, we can use the following bounds f_l, f_u in the algorithm. For a lower bound, set $f_l = s_{min}$, the smallest positive variance coordinate, and for an upper bound take $f_u = nm_{max} + c\sqrt{ns_{max}}$, where m_{max} and s_{max} are the largest coordinates of the mean and variance vectors respectively. Additionally run the linear oracle once with weight vector equal to the vector of means, over the subset of coordinates with zero variances and return that solution if it is better. (In particular, we can solve the problem even if the optimal objective value is zero.) In a continuous problem we need to assume that a positive lower bound f_l is given.

Theorem 3.2.5. *There is an approximation scheme for the nonconvex problem (3.3), which uses an exact oracle for solving the underlying problem (3.2). This algorithm returns a* $(1+\epsilon)$*-approximate solution and makes* $(1 + \frac{2}{\epsilon}\log(\frac{f_u}{f_l}))(1 + \frac{\log(\frac{1}{16\epsilon^2})}{2\log(1+2\epsilon)})$ *oracle queries, namely logarithmic in the size of input and polynomial in* $\frac{1}{\epsilon}$.

■ 3.3 Extension to approximate linear oracles

Suppose we have a δ-approximate linear oracle for solving problem (3.2). We will provide an algorithm for the nonconvex problem (3.3) with approximation factor $\delta(1+\epsilon)$, which invokes the linear oracle a small number of times that is logarithmic in the input-size and polynomial in $\frac{1}{\epsilon}$.

First, we show that if we can guess the optimal linear objective, given by the slope of the tangent to the corresponding level set at the optimal solution, then applying the approximate linear oracle returns an approximate solution with the same multiplicative approximation factor δ. The above statement reduces to showing the following geometric fact.

Lemma 3.3.1. *Consider levels $0 \le \lambda_1 < \lambda_2$ and two parallel lines tangent to the corresponding level sets L_{λ_1} and L_{λ_2} at points (m_1, s_1) and (m_2, s_2) respectively. Further, suppose the corresponding y-intercepts of these lines are b_1 and b_2. Then $\frac{b_2}{b_1} = \frac{\lambda_2+m_2}{\lambda_1+m_1} \ge \frac{\lambda_2}{\lambda_1}$.*

Proof. The function defining a level set L_λ has the form $y = \frac{(\lambda-x)^2}{c^2}$, and so the slope of the tangent to the level set at a point $(m,s) \in L_\lambda$ is given by the first derivative at the point, $-\frac{2(\lambda-x)}{c^2}|_{x=m} = -\frac{2(\lambda-m)}{c^2} = -\frac{2\sqrt{s}}{c}$. Therefore the equation of the tangent line is $y = -\frac{2\sqrt{s}}{c}x + b$, where

$$b = s + \frac{2\sqrt{s}}{c}m = \sqrt{s}(\sqrt{s} + \frac{2m}{c}) = \sqrt{s}(\frac{\lambda - m}{c} + \frac{2m}{c}) = \sqrt{s}(\frac{\lambda+m}{c})$$

Since the two tangents from the lemma are parallel, their slopes are equal: $-\frac{2\sqrt{s_1}}{c} = -\frac{2\sqrt{s_2}}{c}$, therefore $s_1 = s_2$ and equivalently $(\lambda_1 - m_1) = (\lambda_2 - m_2)$.

Therefore the y-intercepts of the two tangents satisfy

$$\frac{b_2}{b_1} = \frac{\sqrt{s_2}(\frac{\lambda_2+m_2}{c})}{\sqrt{s_1}(\frac{\lambda_1+m_1}{c})} = \frac{\lambda_2+m_2}{\lambda_1+m_1} \ge \frac{\lambda_1}{\lambda_2}.$$

The last inequality follows from the fact that $\lambda_2 > \lambda_1$ and $\lambda_1 - m_1 = \lambda_2 - m_2$ (and equality is achieved when $m_1 = \lambda_1$ and $m_2 = \lambda_2$). □

Corollary 3.3.2. *Suppose the optimal solution to the nonconvex problem (3.3) is (m_1, s_1) with objective value λ_1. If we can guess the slope $-a$ of the tangent to the level set L_{λ_1} at the optimal solution, then applying the approximate linear oracle for solving problem (3.2) with respect to that slope will give a δ-approximate solution to problem (3.3).*

Proof. The approximate linear oracle will return a solution (m', s') with value $b_2 = s' + am' \le \delta b_1$, where $b_1 = s_1 + am_1$. The objective function value of (m', s') is at most λ_2, which is the value at the level set tangent to the line $y = -ax + b_2$. By Lemma 3.3.1, $\frac{\lambda_2}{\lambda_1} \le \frac{b_2}{b_1} \le \delta$, therefore the approximation solution has objective function value at most δ times the optimal value, QED. □

If we cannot guess the slope at the optimal solution, we would have to approximate it. The next lemma proves that if we apply the approximate linear oracle to slope that is within $(1 + \sqrt{\frac{\epsilon}{1+\epsilon}})$ of the optimal slope, we would still get a good approximate solution with approximation factor $\delta(1+\epsilon)$.

Lemma 3.3.3. *Consider the level set L_λ and points (m^*, s^*) and (m, s) on it, at which the tangents to L_λ have slopes $-a$ and $-a(1+\sqrt{\frac{\epsilon}{1+\epsilon}})$ respectively. Let the y-intercepts of the tangent line at (m, s) and the line parallel to it through (m^*, s^*) be b_1 and b respectively. Then $\frac{b}{b_1} \leq 1+\epsilon$.*

Proof. Let $\xi = \sqrt{\frac{\epsilon}{1+\epsilon}}$. As established in the proof of Lemma 3.3.1, the slope of the tangent to the level set L_λ at point (m^*, s^*) is $-a = -\frac{2\sqrt{s^*}}{c}$. Similarly the slope of the tangent at (m, s) is $-a(1+\xi) = -\frac{2\sqrt{s}}{c}$. Therefore, $\sqrt{s} = (1+\xi)\sqrt{s^*}$, or equivalently $(\lambda - m) = (1+\xi)(\lambda - m^*)$.

Since b, b_1 are intercepts with the y-axis, of the lines with slopes $-a(1+\xi) = -\frac{2\sqrt{s}}{c}$ containing the points (m^*, s^*), (m, s) respectively, we have

$$\begin{aligned} b_1 &= s + \frac{2\sqrt{s}}{c}m = \frac{\lambda^2 - m^2}{c^2} \\ b &= s^* + (1+\xi)\frac{2\sqrt{s^*}}{c}m^* = \frac{\lambda - m^*}{c^2}(\lambda + m^* + 2\xi m^*). \end{aligned}$$

Therefore

$$\frac{b}{b_1} = \frac{(\lambda - m^*)(\lambda + m^* + 2\xi m^*)}{(\lambda - m)(\lambda + m)} = \frac{1}{1+\xi}\frac{\lambda + m^* + 2\xi m^*}{\lambda + m} \leq \frac{1}{1+\xi}\left(\frac{1}{1-\xi}\right) = \frac{1}{1-\xi^2} = 1+\epsilon,$$

where the last inequality follows by Lemma 3.3.4. □

Lemma 3.3.4. *Following the notation of Lemma 3.3.3, $\frac{\lambda + m^* + 2\xi m^*}{\lambda + m} \leq \frac{1}{1-\xi}$.*

Proof. Recall from the proof of Lemma 3.3.3 that $(\lambda - m) = (1+\xi)(\lambda - m^*)$, therefore $m = \lambda - (1+\xi)(\lambda - m^*) = -\xi\lambda + (1+\xi)m^*$. Hence,

$$\frac{\lambda + m^* + 2\xi m^*}{\lambda + m} = \frac{\lambda + (1+2\xi)m^*}{(1-\xi)\lambda + (1+\xi)m^*} = \frac{\frac{\lambda}{m^*} + (1+2\xi)}{(1-\xi)\frac{\lambda}{m^*} + (1+\xi)} \leq \frac{1}{1-\xi},$$

since $\frac{1+2\xi}{1+\xi} \leq \frac{1}{1-\xi}$ for $\xi \in [0, 1)$. □

A corollary from Lemma 3.3.1 and Lemma 3.3.3 is that applying the linear oracle with respect to slope that is within $(1+\sqrt{\frac{\epsilon}{1+\epsilon}})$ times of the optimal slope yields an approximate solution with objective value within $(1+\epsilon)\delta$ times of the optimal.

Lemma 3.3.5. *Suppose the optimal solution to the nonconvex problem (3.3) is (m^*, s^*) with objective value λ and the slope of the tangent to the level set L_λ at it is $-a$. Then running the δ-approximate oracle for solving problem (3.2) with respect to slope that is in $[-a, -a(1+\sqrt{\frac{\epsilon}{1+\epsilon}})]$ returns a solution to (3.3) with objective function value no greater than $(1+\epsilon)\delta\lambda$.*

Proof. Suppose the optimal solution with respect to the linear objective specified by slope $-a(1+\sqrt{\frac{\epsilon}{1+\epsilon}})$ has value $b' \in [b_1, b]$, where b_1, b are the y-intercepts of the lines with that slope, tangent to L_λ and passing through (m^*, s^*) respectively (See Figure 3.3-right). Then applying the δ-approximate linear oracle to the same linear objective returns solution with value $b_2 \geq \delta b'$. Hence $\frac{b_2}{b} \leq \frac{b_2}{b'} \leq \delta$.

On the other hand, the approximate solution returned by the linear oracle has value of our original objective function equal to at most λ_2, where L_{λ_2} is the level set tangent to the line on which the approximate solution lies. By Lemma 3.3.1, $\frac{\lambda_2}{\lambda} \leq \frac{b_2}{b_1} = \frac{b_2}{b}\frac{b}{b_1} \leq \delta(1+\epsilon)$, where the last inequality follows by Lemma 3.3.3 and the above bound on $\frac{b_2}{b}$. □

Finally, we are ready to state our theorem for solving the concave minimization problem (3.3). The theorem says that there is an algorithm for this problem with essentially the same approximation factor as the underlying problem (3.2), which makes only logarithmically many calls to the latter.

Theorem 3.3.6. *Suppose we have a δ-approximate oracle for solving problem (3.2). The nonconvex problem (3.3) can be approximated to a multiplicative factor of $\delta(1+\epsilon)$ by calling the oracle logarithmically many times in the input size and polynomially many times in $\frac{1}{\epsilon}$.*

Proof. We use the same type of algorithm as in Theorem 3.4.3: apply the available approximate linear oracle on a geometric progression of weight vectors (slopes), determined by the lemmas above. In particular, apply it to slopes $U, (1+\xi)U, ..., (1+\xi)^i U = L$, where $\xi = \sqrt{\frac{\epsilon}{1+\epsilon}}$, L is a lower bound for the optimal slope and U is an upper bound for it. For each approximate feasible solution obtained, compute its objective function value and return the solution with minimum objective function value. By Lemma 3.3.5, the value of the returned solution would be within $\delta(1+\epsilon)$ of the optimal.

Note that it is possible for the optimal slope to be 0: this would happen when the optimal solution satisfies $m^* = \lambda$ and $s^* = 0$. We have to handle this case differently: run the linear oracle just over the subset of coordinates with zero variance-values, to find the approximate solution with smallest m. Return this solution if its value is better than the best solution among the above.

It remains to bound the values L and U. We established earlier that the optimal slope is given by $\frac{2\sqrt{s^*}}{c}$, where s^* is the variance of the optimal solution. Among the solutions with nonzero variance, the variance of a feasible solution is at least s_{min}, the smallest possible nonzero variance of a single element, and at most $(\lambda_{max})^2 \leq (nm_{max} + c\sqrt{ns_{max}})^2$, where m_{max} is the largest possible mean of a single element and s_{max} is the largest possible variance of a single element (assuming that a feasible solution uses each element in the ground set at most once). Thus, set $U = -\frac{2\sqrt{s_{min}}}{c}$ and $L = -\frac{2(nm_{max}+c\sqrt{ns_{max}})}{c}$ □

3.4 Extension to other objectives with exact and approximate linear oracles

We now consider other stochastic models with risk, which incorporate the probability distribution tails in the objective function or constraints. Following the notation from the previous sections, recall that $\mathbf{W} = (W_1, ..., W_n)$ is the vector of stochastic weights, with corresponding means $\boldsymbol{\mu} = (\mu_1, ..., \mu_n)$ and variances $\boldsymbol{\tau} = (\tau_1, ..., \tau_n)$.

3.4.1 Probability tail objective

One natural objective would be to pick an object whose weight does not exceed a given budget or threshold (*e.g.*, when going to the airport, we would like a route whose traversal does not exceed the time our flight departs). In a stochastic setting, we would therefore want the object with maximum probability that its stochastic weight does not exceed the threshold t. Equivalently, the user penalty or cost is a step function equal to 1 when the weight of the solution exceeds the threshold and 0 otherwise.

When the weights W_i come from independent normal distributions, a feasible solution $\mathbf{x}$ will have a normally distributed weight $\mathbf{W}^T\mathbf{x} \sim N(\mu^T\mathbf{x}, \tau^T\mathbf{x})$. Therefore, the probability that the weight does not exceed the threshold is

$$\mathbf{Pr}\left[\mathbf{W}^T\mathbf{x} \leq t\right] = \mathbf{Pr}\left[\frac{\mathbf{W}^T\mathbf{x} - \mu^T\mathbf{x}}{\sqrt{\tau^T\mathbf{x}}} \leq \frac{t - \mu^T\mathbf{x}}{\sqrt{\tau^T\mathbf{x}}}\right] = \Phi\Big(\frac{t - \mu^T\mathbf{x}}{\sqrt{\tau^T\mathbf{x}}}\Big),$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal random variable $N(0,1)$. Since $\Phi(\cdot)$ is monotone increasing, maximizing the probability tail objective $\mathbf{Pr}\left[\mathbf{W}^T\mathbf{x} \leq t\right]$ is equivalent to the following quasiconvex maximization problem:

$$\begin{array}{ll} \text{maximize} & \dfrac{t - \mu^T\mathbf{x}}{\sqrt{\tau^T\mathbf{x}}} \\ \text{subject to} & \mathbf{x} \in \mathcal{F}. \end{array} \tag{3.4}$$

By Lemma 3.4.1 below we have that an approximation for the nonconvex problem (3.4) yields the same approximation factor for the stochastic problem.[3]

Lemma 3.4.1. *A δ-approximation for the nonconvex threshold objective (3.4) yields a δ-approximation for the stochastic threshold objective $\Phi\left(\frac{t-\mu^T\mathbf{x}}{\sqrt{\tau^T\mathbf{x}}}\right)$, where Φ denotes the cumulative distribution function of the standard normal random variable $N(0,1)$.*

Proof. Denote the approximate and the optimal solutions by $\mathbf{x}$, $\mathbf{x}_{opt}$ respectively. A δ-approximation for maximizing the nonconvex threshold objective means that

$$\frac{t - \mu^T\mathbf{x}}{\sqrt{\tau^T\mathbf{x}}} \geq \delta \frac{t - \mu^T\mathbf{x}_{opt}}{\sqrt{\tau^T\mathbf{x}_{opt}}}.$$

Denote $f = \frac{t-\mu^T\mathbf{x}}{\sqrt{\tau^T\mathbf{x}}}$ and $f_{opt} = \frac{t-\mu^T\mathbf{x}_{opt}}{\sqrt{\tau^T\mathbf{x}_{opt}}}$. Since by assumption, $f_{opt} \geq 0$, a line going through 0 and between the points $(f, \Phi(f))$, $(f_{opt}, \Phi(f_{opt}))$ on the graph of the function Φ will cross the vertical lines through this points below the graph and above the graph respectively (at the points A' and B' in Figure 3.4). Using the notation from Figure 3.4, therefore $\Phi(f) \geq y$ (the y-coordinate of

[3]We expect that under reasonable conditions, *e.g.*, if a feasible solution $\mathbf{x}$ has sufficiently many nonzero coordinates, arbitrary weight distributions will lead to feasible solutions having approximately normal weight by the Central Limit Theorem. Thus our algorithms are likely to provide a good approximation in that general case as well.

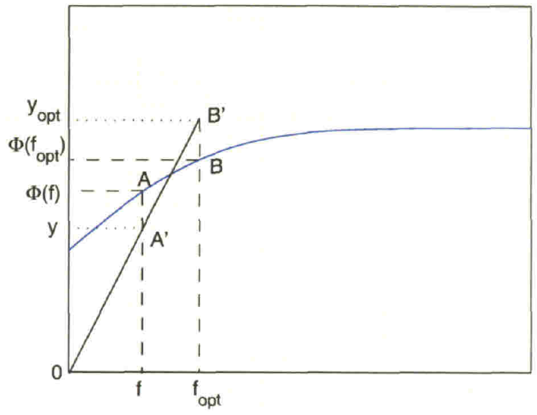


Figure 3.4. A plot of the stochastic threshold objective (the cumulative distribution function Φ of the standard normal random variable).

A') and $\Phi(f_{opt}) \leq y_{opt}$. On the other hand, since the lines AA' and BB' are parallel, we have the equality below:

$$\delta \leq \frac{f}{f_{opt}} = \frac{y}{y_{opt}} \leq \frac{\Phi(f)}{\Phi(f_{opt})},$$

therefore $\Phi(f) \geq \delta\Phi(f_{opt})$, QED. □

Although this model may be viewed as more restrictive in that it assumes normal distributions as opposed to arbitrary ones in our previous objective, it is mathematically more difficult because of the unbounded gradient and the fact that the function can attain both positive, zero and negative values. Thus it is not even clear that one can obtain a multiplicative approximation (imagine there were only one solution with positive objective value and all others are negative or zero). We show in Section 3.5 that when the maximum is positive or zero (that is, there is at least one feasible point whose mean is within the threshold t), we can obtain an arbitrarily good approximation with exact linear oracles; when it is negative, the problem is still open.

The analogues of the algorithms from Sections 3.2 and 3.3 provide the following approximations here.

Theorem 3.4.2. *Suppose we have an exact oracle for problem (3.2) and suppose the smallest mean feasible solution satisfies $\boldsymbol{\mu}^T\mathbf{x} \leq t$. Then for any $\epsilon \in (0,1)$, there is an $(1-\epsilon)$-approximate algorithm for solving the nonconvex threshold problem (3.4), which makes a logarithmic in the input and polynomial in $\frac{1}{\epsilon}$ number of oracle calls.*

The following extends our results to the case where we only have an approximate linear oracle.

Theorem 3.4.3. *Suppose we have a δ-approximation linear oracle for problem (3.2). Then, the nonconvex threshold problem (3.4) has a $\sqrt{1-\left[\frac{\delta-(1-\epsilon^2/4)}{(2+\epsilon)\epsilon/4}\right]}$-approximation algorithm that calls the linear oracle a logarithmic in the input and polynomial in $\frac{1}{\epsilon}$ number of times, assuming the optimal solution to (3.4) satisfies $\boldsymbol{\mu}^T\mathbf{x}^* \leq (1-\epsilon)t$.*

3.4.2 Value-at-risk objective

Another popular model in portfolio optimization is the value-at-risk objective (for example, see Ghaoi *et al.* [51]), which in our context of stochastic optimization following the above notation can be defined as follows:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \mathbf{Pr}(\mathbf{W}^T\mathbf{x} \leq t) \geq p \\ & \mathbf{x} \in \mathcal{F}, \end{array} \tag{3.5}$$

for given probability (confidence level) p.

When the element weights come from independent normal distributions, this model has the equivalent nonconvex formulation

$$\begin{array}{ll} \text{minimize} & \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}} \\ \text{subject to} & \mathbf{x} \in \mathcal{F} \end{array}$$

where $c = \Phi^{-1}(p)$ and as before $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal random variable $N(0,1)$, and $\Phi^{-1}(\cdot)$ denotes its inverse.

For arbitrary distributions, we can apply Chebyshev's bound $\mathbf{Pr}(\mathbf{W}^T\mathbf{x} \geq \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}}) \leq \frac{1}{c^2}$, or equivalently $\mathbf{Pr}(\mathbf{W}^T\mathbf{x} < \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}}) > 1 - \frac{1}{c^2}$. Taking $c = \frac{1}{\sqrt{1-p}}$ gives the inequality $\mathbf{Pr}(\mathbf{W}^T\mathbf{x} < \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}}) > p$. This shows the following lemma:

Lemma 3.4.4. *The optimal value of the nonconvex problem*

$$\begin{array}{ll} \textit{minimize} & \boldsymbol{\mu}^T\mathbf{x} + \dfrac{1}{\sqrt{1-p}}\sqrt{\boldsymbol{\tau}^T\mathbf{x}} \\ \textit{subject to} & \mathbf{x} \in \mathcal{F} \end{array}$$

will provide an upper bound for the minimum threshold t in the value-at-risk problem with given probability p.

We remark that in the absence of more information on the distributions, other than their means and standard deviations, this is the best one can do (assuming Chebyshev's bound is the best available bound) to solve the value-at-risk problem.

For an illustration of the difference between the normal and arbitrary distributions, consider again the following hypothetical shortest path problem.

Example 3.4.5. *Suppose we need to reach the airport by a certain time. We want to find the minimum time (and route) that we need to allocate for our trip so as to arrive on time with probability at least $p = .95$. (That is, how close can we cut it to the deadline and not be late?) If we know that the travel times on the edges are normally distributed, the minimum time equals $\min_{\mathbf{x}\in\mathcal{F}} \boldsymbol{\mu}^T\mathbf{x} + 1.645\sqrt{\boldsymbol{\tau}^T\mathbf{x}}$, since $\Phi^{-1}(.95) = 1.645$. On the other hand, if we had no information about the distributions, we should instead allocate the upper bound $\min_{\mathbf{x}\in\mathcal{F}} \boldsymbol{\mu}^T\mathbf{x} + 4.5\sqrt{\boldsymbol{\tau}^T\mathbf{x}}$, since $\frac{1}{\sqrt{1-0.95}} \approx 4.5$ (which still guarantees that we would arrive with probability at least 95%).*

■ 3.4.2 Value-at-risk objective

Another popular model in portfolio optimization is the value-at-risk objective (for example, see Ghaoi *et al.* [51]), which in our context of stochastic optimization following the above notation can be defined as follows:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \mathbf{Pr}(\mathbf{W}^T\mathbf{x} \le t) \ge p \\ & \mathbf{x} \in \mathcal{F}, \end{array} \tag{3.5}$$

for given probability (confidence level) p.

When the element weights come from independent normal distributions, this model has the equivalent nonconvex formulation

$$\begin{array}{ll} \text{minimize} & \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}} \\ \text{subject to} & \mathbf{x} \in \mathcal{F} \end{array}$$

where $c = \Phi^{-1}(p)$ and as before $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal random variable $N(0,1)$, and $\Phi^{-1}(\cdot)$ denotes its inverse.

For arbitrary distributions, we can apply Chebyshev's bound $\mathbf{Pr}(\mathbf{W}^T\mathbf{x} \ge \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}}) \le \frac{1}{c^2}$, or equivalently $\mathbf{Pr}(\mathbf{W}^T\mathbf{x} < \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}}) > 1 - \frac{1}{c^2}$. Taking $c = \frac{1}{\sqrt{1-p}}$ gives the inequality $\mathbf{Pr}(\mathbf{W}^T\mathbf{x} < \boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}}) > p$. This shows the following lemma:

Lemma 3.4.4. *The optimal value of the nonconvex problem*

$$\begin{array}{ll} \textit{minimize} & \boldsymbol{\mu}^T\mathbf{x} + \dfrac{1}{\sqrt{1-p}}\sqrt{\boldsymbol{\tau}^T\mathbf{x}} \\ \textit{subject to} & \mathbf{x} \in \mathcal{F} \end{array}$$

will provide an upper bound for the minimum threshold t in the value-at-risk problem with given probability p.

We remark that in the absence of more information on the distributions, other than their means and standard deviations, this is the best one can do (assuming Chebyshev's bound is the best available bound) to solve the value-at-risk problem.

For an illustration of the difference between the normal and arbitrary distributions, consider again the following hypothetical shortest path problem.

Example 3.4.5. *Suppose we need to reach the airport by a certain time. We want to find the minimum time (and route) that we need to allocate for our trip so as to arrive on time with probability at least $p = .95$. (That is, how close can we cut it to the deadline and not be late?) If we know that the travel times on the edges are normally distributed, the minimum time equals $\min_{\mathbf{x}\in\mathcal{F}} \boldsymbol{\mu}^T\mathbf{x} + 1.645\sqrt{\boldsymbol{\tau}^T\mathbf{x}}$, since $\Phi^{-1}(.95) = 1.645$. On the other hand, if we had no information about the distributions, we should instead allocate the upper bound $\min_{\mathbf{x}\in\mathcal{F}} \boldsymbol{\mu}^T\mathbf{x} + 4.5\sqrt{\boldsymbol{\tau}^T\mathbf{x}}$, since $\frac{1}{\sqrt{1-0.95}} \approx 4.5$ (which still guarantees that we would arrive with probability at least 95%).*

■ 3.5 Proofs for the probability tail objective with exact linear oracle

In this section, we give an approximation scheme for the nonconvex problem formulation (3.4) that uses access to an exact oracle for solving the linear problem (3.2).

The algorithms are analogous to the ones for our nonconvex objective from Sections 3.2 and 3.3; however, the analysis in the latter does not apply directly because the geometric lemmas are tailored to the specific function form. Thus, for completeness we provide a self-contained analysis for the nonconvex probability tail objective here.

Our algorithms assume that the maximum of the objective is non-negative, in other words the feasible solution with smallest mean satisfies $\boldsymbol{\mu}^T\mathbf{x} \leq t$. Note, it is not clear a priori that that such a multiplicative approximation is even possible, since we still let the function have positive, zero or negative values on different feasible solutions. The case in which the maximum is negative is structurally very different (the objective on its negative range no longer attains optima at extreme points) and remains open. Even with this assumption, approximating the objective function is especially challenging due to its unbounded gradient and the form of its numerator.

We first note that if the optimal solution has variance 0, we can find it exactly with a single oracle query: Apply the linear oracle on the set of elements with zero variances to find the feasible solution with smallest mean. If the mean is no greater than t, output the solution, otherwise conclude that the optimal solution has positive variance and proceed with the approximation scheme below.

The main technical lemma that our algorithm is based on is an extension of the concept of separation and optimization: instead of deciding whether a line (hyperplane) is separating for a convex set, in the sense that the set lies entirely on one side of the line (hyperplane), we construct an approximate oracle which decides whether a non-linear curve (in our case, a parabola) is separating for the convex set.

From here on we will analyze the projections of the objective function and the feasible set onto the plane $span(\boldsymbol{\mu}, \boldsymbol{\tau})$ since the nonconvex problem (3.4) is equivalent in that space. Consider the lower level sets $\underline{L}_\lambda = \{\mathbf{z} \mid f(\mathbf{z}) \leq \lambda\}$ of the objective function $f(m,s) = \frac{t-m}{\sqrt{s}}$, where $m, s \in \mathbb{R}$. Denote $L_\lambda = \{\mathbf{z} \mid f(\mathbf{z}) = \lambda\}$. We first prove that any level set boundary can be approximated by a small number of linear segments. The main work here involves deriving a condition for a linear segment with endpoints on L_λ, to have objective function values within $(1-\epsilon)$ of λ.

Lemma 3.5.1. *Consider the points* $(m_1, s_1), (m_2, s_2) \in L_\lambda$ *with* $s_1 > s_2 > 0$. *The segment connecting these two points is contained in the level set region* $\underline{L}_\lambda \backslash \underline{L}_{\lambda(1-\epsilon)}$ *whenever* $s_2 \geq (1-\epsilon)^4 s_1$, *for every* $\epsilon \in (0,1)$.

Proof. Any point on the segment $[(m_1, s_1), (m_2, s_2)]$ can be written as a convex combination of its endpoints, $(\alpha m_1 + (1-\alpha)m_2, \alpha s_1 + (1-\alpha)s_2)$, where $\alpha \in [0,1]$. Consider the function $h(\alpha) = f(\alpha m_1 + (1-\alpha)m_2, \alpha s_1 + (1-\alpha)s_2)$. We have,

$$h(\alpha) = \frac{t - \alpha m_1 - (1-\alpha)m_2}{\sqrt{\alpha s_1 + (1-\alpha)s_2}} = \frac{t - \alpha(m_1 - m_2) - m_2}{\sqrt{\alpha(s_1 - s_2) + s_2}}$$

We want to find the point on the segment with smallest objective value, so we minimize with respect to α.

$$\begin{aligned} h'(\alpha) &= \frac{(m_2 - m_1)\sqrt{\alpha(s_1 - s_2) + s_2} - [t - \alpha(m_1 - m_2) - m_2] * \frac{1}{2}(s_1 - s_2)/\sqrt{\alpha(s_1 - s_2) + s_2}}{\alpha(s_1 - s_2) + s_2} \\ &= \frac{2(m_2 - m_1)[\alpha(s_1 - s_2) + s_2] - [t - \alpha(m_1 - m_2) - m_2](s_1 - s_2)}{2[\alpha(s_1 - s_2) + s_2]^{3/2}} \\ &= \frac{\alpha(m_2 - m_1)(s_1 - s_2) + 2(m_2 - m_1)s_2 - (t - m_2)(s_1 - s_2)}{2[\alpha(s_1 - s_2) + s_2]^{3/2}}. \end{aligned}$$

Setting the derivative to 0 is equivalent to setting the numerator above to 0, thus we get

$$\alpha_{\min} = \frac{(t - m_2)(s_1 - s_2) - 2(m_2 - m_1)s_2}{(m_2 - m_1)(s_1 - s_2)} = \frac{t - m_2}{m_2 - m_1} - \frac{2s_2}{s_1 - s_2}.$$

Note that the denominator of $h'(\alpha)$ is positive and its numerator is linear in α, with a positive slope, therefore the derivative is negative for $\alpha < \alpha_{\min}$ and positive otherwise, so $\alpha_{\min}$ is indeed a global minimum as desired.

It remains to verify that $h(\alpha_{\min}) \geq (1 - \epsilon)\lambda$. Note that $t - m_i = \lambda\sqrt{s_i}$ for $i = 1, 2$ since $(m_i, s_i) \in L_\lambda$ and consequently, $m_2 - m_1 = \lambda(\sqrt{s_1} - \sqrt{s_2})$. We use this further down in the following expansion of $h(\alpha_{\min})$.

$$\begin{aligned} h(\alpha_{\min}) &= \frac{t + \alpha_{\min}(m_2 - m_1) - m_2}{\sqrt{\alpha_{\min}(s_1 - s_2) + s_2}} = \frac{t + (\frac{t - m_2}{m_2 - m_1} - \frac{2s_2}{s_1 - s_2})(m_2 - m_1) - m_2}{\sqrt{(\frac{t - m_2}{m_2 - m_1} - \frac{2s_2}{s_1 - s_2})(s_1 - s_2) + s_2}} \\ &= \frac{t + t - m_2 - 2s_2\frac{m_2 - m_1}{s_1 - s_2} - m_2}{\sqrt{(t - m_2)\frac{s_1 - s_2}{m_2 - m_1} - 2s_2 + s_2}} = \frac{2(t - m_2) - 2s_2\frac{\lambda(\sqrt{s_1} - \sqrt{s_2})}{s_1 - s_2}}{\sqrt{\lambda\sqrt{s_2}\frac{s_1 - s_2}{\lambda(\sqrt{s_1} - \sqrt{s_2})} - s_2}} \\ &= \frac{2\lambda\sqrt{s_2} - 2s_2\frac{\lambda}{\sqrt{s_1} + \sqrt{s_2}}}{\sqrt{\sqrt{s_2}(\sqrt{s_1} + \sqrt{s_2}) - s_2}} = 2\lambda\frac{\sqrt{s_2} - \frac{s_2}{\sqrt{s_1} + \sqrt{s_2}}}{\sqrt{\sqrt{s_1 s_2}}} \\ &= 2\lambda\frac{\sqrt{s_1 s_2} + s_2 - s_2}{(s_1 s_2)^{1/4}(\sqrt{s_1} + \sqrt{s_2})} = 2\lambda\frac{(s_1 s_2)^{1/4}}{\sqrt{s_1} + \sqrt{s_2}}. \end{aligned}$$

We need to show that when the ratio s_1/s_2 is sufficiently close to 1, $h(\alpha_{\min}) \geq (1 - \epsilon)\lambda$, or equivalently

$$\frac{2(s_1 s_2)^{1/4}}{\sqrt{s_1} + \sqrt{s_2}} \geq 1 - \epsilon \qquad \Leftrightarrow \qquad 2(s_1 s_2)^{1/4} \geq (1 - \epsilon)(s_1^{1/2} + s_2^{1/2})$$

$$\Leftrightarrow \quad (1 - \epsilon)\Big(\frac{s_1}{s_2}\Big)^{1/2} - 2\Big(\frac{s_1}{s_2}\Big)^{1/4} + (1 - \epsilon) \leq 0 \tag{3.6}$$

The minimum of the last quadratic function above is attained at $\left(\frac{s_1}{s_2}\right)^{1/4} = \frac{1}{1-\epsilon}$ and we can check that at this minimum the quadratic function is indeed negative:

$$(1 - \epsilon)\Big(\frac{1}{1 - \epsilon}\Big)^2 - 2\Big(\frac{1}{1 - \epsilon}\Big) + (1 - \epsilon) = (1 - \epsilon) - \frac{1}{1 - \epsilon} < 0,$$

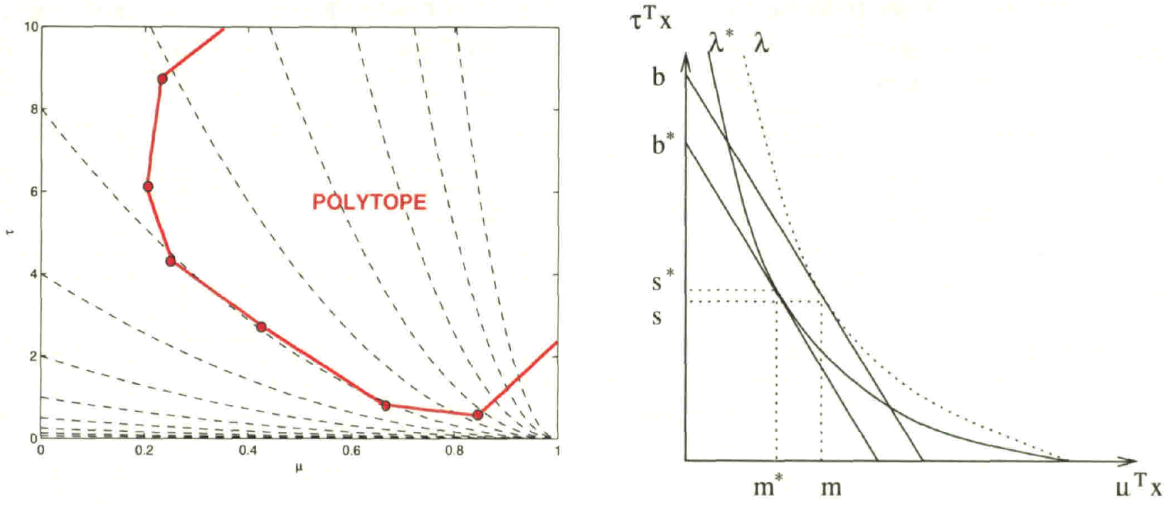


Figure 3.5. *(a)* Level sets of the objective function and the convex hull of the projected feasible set on the $span(\boldsymbol{\mu}, \boldsymbol{\tau})$-plane. *(b)* Applying the approximate linear oracle on the optimal slope gives an approximate value b of the corresponding linear objective value b^*. The resulting solution has nonlinear objective function value of at least λ, which is an equally good approximation for the optimal value λ^*.

for all $0 < \epsilon < 1$. The inequality (3.6) is satisfied at $\frac{s_1}{s_2} = 1$, therefore it holds for all $\left(\frac{s_1}{s_2}\right) \in [1, \frac{1}{(1-\epsilon)^4}]$. Hence, a sufficient condition for $h(\alpha_{\min}) \le (1-\epsilon)\lambda$ is $s_2 \ge (1-\epsilon)^4 s_1$, and we are done. □

Lemma 3.5.1 now makes it easy to show our main lemma, namely that any level set L_λ can be approximated within a multiplicative factor of $(1-\epsilon)$ via a small number of segments. Let s_{min} and s_{max} be a lower and upper bound respectively for the variance of the optimal solution. For example, take s_{min} to be the smallest positive coordinate of the variance vector, and s_{max} the variance of the feasible solution with smallest mean.

Lemma 3.5.2. *The level set* $L_\lambda = \{(m, s) \in \mathbb{R}^2 \mid \frac{t-m}{\sqrt{s}} = \lambda\}$ *can be approximated within a factor of* $(1-\epsilon)$ *by* $\left\lceil \frac{1}{4}\log\left(\frac{s_{\max}}{s_{\min}}\right) / \log\frac{1}{1-\epsilon}\right\rceil$ *linear segments.*

Proof. By definition of s_{min} and s_{max}, the the variance of the optimal solution ranges from $s_{\min}$ to $s_{\max}$. By Lemma 3.5.1, the segments connecting the points on L_λ with variances $s_{\max}, s_{\max}(1-\epsilon)^4, s_{\max}(1-\epsilon)^8, ..., s_{\min}$ all lie in the level set region $\underline{L}_\lambda \backslash \underline{L}_{\lambda(1-\epsilon)}$, that is they underestimate and approximate the level set L_λ within a factor of $(1-\epsilon)$. The number of these segments is $\lceil \frac{1}{4}\log\left(\frac{s_{\max}}{s_{\min}}\right) / \log\frac{1}{1-\epsilon}\rceil$. □

The above lemma yields an approximate separation oracle for the nonlinear level set L_λ and polytope($\mathcal{F}$). The oracle takes as input the level λ and either returns a solution $\mathbf{x}$ with objective value $f(\mathbf{x}) \ge (1-\epsilon)\lambda$ from the feasible set, or guarantees that $f(\mathbf{x}) < \lambda$ for all $\mathbf{x} \in$ polytope($\mathcal{F}$). Therefore, an exact oracle for solving problem (3.2) allows us to obtain an approximate nonlinear separation oracle, by applying the former to weight vectors $a\boldsymbol{\mu} + \boldsymbol{\tau}$, for all possible slopes $(-a)$ of the segments approximating the level set. We formalize this in the next theorem.

Theorem 3.5.3 (Approximate Nonlinear Separation Oracle). *Suppose we have an exact (linear) oracle for solving problem (3.2). Then, we can construct a nonlinear oracle which solves the following approximate separation problem: given a level λ and $\epsilon \in (0,1)$, the oracle returns*

1. *A solution* $\mathbf{x} \in \mathcal{F}$ *with* $f(\mathbf{x}) \geq (1-\epsilon)\lambda$, or
2. *An answer that* $f(\mathbf{x}) < \lambda$ *for all* $\mathbf{x} \in$ *polytope*$(\mathcal{F})$,

and the number of linear oracle calls it makes is $\frac{1}{4}\log\left(\frac{s_{\max}}{s_{\min}}\right)/\log\frac{1}{1-\epsilon}$, *that is* $O(\frac{1}{\epsilon}\log\frac{s_{\max}}{s_{\min}})$.

We can now give an approximation algorithm for the nonconvex problem (3.4), by applying the above nonlinear oracle on a geometric progression of possible values λ of the objective function f. We first need to bound the maximum value f_{opt} of the objective function f. A lower bound f_l is provided by the solution $\mathbf{x}_{mean}$ with smallest mean or the solution $\mathbf{x}_{var}$ with smallest positive variance, whichever has a higher objective value: $f_l = \max\{f(\mathbf{x}_{mean}), f(\mathbf{x}_{var})\}$ where $f(\mathbf{x}) = \frac{t-\boldsymbol{\mu}^T\mathbf{x}}{\sqrt{\boldsymbol{\tau}^T\mathbf{x}}}$. On the other hand, $\boldsymbol{\mu}^T\mathbf{x} \geq \boldsymbol{\mu}^T\mathbf{x}_{mean}$ and $\boldsymbol{\tau}^T\mathbf{x} \geq \boldsymbol{\tau}^T\mathbf{x}_{var}$ for all $\mathbf{x} \in$ polytope$(\mathcal{F})$, so an upper bound for the objective f is given by $f_u = \frac{t-\boldsymbol{\mu}^T\mathbf{x}_{mean}}{\sqrt{\boldsymbol{\tau}^T\mathbf{x}_{var}}}$ (recall that $t - \boldsymbol{\mu}^T\mathbf{x}_{mean} > 0$ by assumption).

Theorem 3.5.4 (Theorem 3.4.2 from main body.). *Suppose we have an exact oracle for problem (3.2) and suppose the smallest mean feasible solution satisfies* $\boldsymbol{\mu}^T\mathbf{x} \leq t$. *Then for any* $\epsilon \in (0,1)$, *there is an algorithm for solving the nonconvex threshold problem (3.4), which returns a feasible solution* $\mathbf{x} \in \mathcal{F}$ *with value at least* $(1-\epsilon)$ *times the optimum, and makes* $O\Big(\log\left(\frac{s_{\max}}{s_{\min}}\right)\log\left(\frac{f_u}{f_l}\right)\frac{1}{\epsilon^2}\Big)$ *oracle calls.*

Proof. Now, apply the approximate separation oracle from Theorem 3.5.3 with $\epsilon' = 1 - \sqrt{1-\epsilon}$ successively on the levels $f_u, (1-\epsilon')f_u, (1-\epsilon')^2 f_u, \ldots$ until we reach a level $\lambda = (1-\epsilon')^i f_u \geq f_l$ for which the oracle returns a feasible solution $\mathbf{x}'$ with

$$f(\mathbf{x}') \geq (1-\epsilon')\lambda = (\sqrt{1-\epsilon})^{i+1} f_u.$$

From running the oracle on the previous level $f_u(1-\epsilon')^{i-1}$, we know that $f(\mathbf{x}) \leq f(\mathbf{x}_{opt}) < (\sqrt{1-\epsilon})^{i-1} f_u$ for all $\mathbf{x} \in$ polytope$(\mathcal{F})$, where $\mathbf{x}_{opt}$ denotes the optimal solution. Therefore,

$$(\sqrt{1-\epsilon})^{i+1} f_u \leq f(\mathbf{x}') \leq f(\mathbf{x}_{opt}) < (\sqrt{1-\epsilon})^{i-1} f_u, \qquad \text{and hence}$$

$$(1-\epsilon) f(\mathbf{x}_{opt}) < f(\mathbf{x}') \leq f(\mathbf{x}_{opt}).$$

So the solution $\mathbf{x}'$ gives a $(1-\epsilon)$-approximation to the optimum $\mathbf{x}_{opt}$. In the process, we run the approximate nonlinear separation oracle at most $\log\left(\frac{f_u}{f_l}\right)/\log\frac{1}{1-\epsilon'}$ times, and each run makes $\frac{1}{4}\log\left(\frac{s_{\max}}{s_{\min}}\right)/\log\frac{1}{1-\epsilon'}$ queries to the linear oracle, so the algorithm makes at most
$\frac{1}{4}\log\left(\frac{s_{\max}}{s_{\min}}\right)\log\left(\frac{f_u}{f_l}\right)/\left(\frac{1}{2}\log\frac{1}{1-\epsilon}\right)^2$ queries to the oracle for the linear problem (3.2). Finally, since $\log\frac{1}{1-\epsilon} \geq \epsilon$ for $\epsilon \in [0,1)$, we get the desired bound for the total number of queries. $\square$

■ 3.6 Proofs for the probability tail objective with approximate linear oracle

In this section, we show that a δ-approximate oracle to problem (3.2) yields an efficient approximation algorithm to the nonconvex problem (3.4). As in Section 3.5, we first check whether the optimal solution has zero variance and if not, proceed with the algorithm and analysis below.

We first prove several geometric lemmas that enable us to derive the approximation guarantees later.

Lemma 3.6.1 (Geometric lemma). *Consider two objective function values $\lambda^* > \lambda$ and points $(m^*, s^*) \in L_{\lambda^*}$, $(m, s) \in L_\lambda$ in the positive orthant such that the tangents to the points at the corresponding level sets are parallel. Then, the y-intercepts b^*, b of the two tangent lines satisfy*

$$b - b^* = s^*\left[1 - \left(\frac{\lambda}{\lambda^*}\right)^2\right].$$

Proof. Suppose the slope of the tangents is $(-a)$, where $a > 0$. Then the y-intercepts of the two tangent lines satisfy

$$b = s + am, \qquad b^* = s^* + am^*.$$

In addition, since the points (m, s) and (m^*, s^*) lie on the level sets L_λ, L_{λ^*}, they satisfy

$$t - m = \lambda\sqrt{s}, \qquad t - m^* = \lambda^*\sqrt{s^*}.$$

Since the first line is tangent at (m, s) to the parabola $y = (\frac{t-x}{\lambda})^2$, the slope equals the first derivative at this point, $-\frac{2(t-x)}{\lambda^2}|_{x=m} = -\frac{2(t-m)}{\lambda^2} = -\frac{2\lambda\sqrt{s}}{\lambda^2} = -\frac{2\sqrt{s}}{\lambda}$, so the absolute value of the slope is $a = \frac{2\sqrt{s}}{\lambda}$. Similarly the absolute value of the slope also satisfies $a = \frac{2\sqrt{s^*}}{\lambda^*}$, therefore

$$\sqrt{s^*} = \frac{\lambda^*}{\lambda}\sqrt{s}.$$

Note that for $\lambda^* > \lambda$, this means that $s^* > s$. From here, we can represent the difference $m - m^*$ as

$$m - m^* = (t - m^*) - (t - m) = \lambda^*\sqrt{s^*} - \lambda\sqrt{s} = \frac{(\lambda^*)^2}{\lambda}\sqrt{s} - \lambda\sqrt{s} = \left[\left(\frac{\lambda^*}{\lambda}\right)^2 - 1\right]\lambda\sqrt{s}.$$

Substituting the slope $a = \frac{2\sqrt{s}}{\lambda}$ in the tangent line equations, we get

$$\begin{aligned}
b - b^* &= s + \frac{2\sqrt{s}}{\lambda}m - s^* - \frac{2\sqrt{s}}{\lambda}m^* \\
&= s - \left(\frac{\lambda^*}{\lambda}\right)^2 s + \frac{2\sqrt{s}}{\lambda}(m - m^*) \\
&= s - \left(\frac{\lambda^*}{\lambda}\right)^2 s + \frac{2\sqrt{s}}{\lambda}\lambda\sqrt{s}\left[\left(\frac{\lambda^*}{\lambda}\right)^2 - 1\right] \\
&= s - \left(\frac{\lambda^*}{\lambda}\right)^2 s + 2s\left[\left(\frac{\lambda^*}{\lambda}\right)^2 - 1\right] \\
&= s\left[\left(\frac{\lambda^*}{\lambda}\right)^2 - 1\right] = s^*\left[1 - \left(\frac{\lambda}{\lambda^*}\right)^2\right],
\end{aligned}$$

as desired. □

The next lemma builds intuition, as well as helps in the analysis of the algorithm. It shows that we can approximate the optimal solution well if we know the optimal weight vector to use with the available approximate oracle for problem (3.2).

Lemma 3.6.2. *Suppose we have a δ-approximate linear oracle for optimizing over the feasible set $\mathcal{F}$ and suppose that the optimal solution satisfies $\mu^T x^* \leq (1-\epsilon)t$. If we can guess the slope of the tangent to the corresponding level set at the optimal point x^*, then we can find a $\sqrt{1-\delta\frac{2-\epsilon}{\epsilon}}$-approximate solution to the nonconvex problem (3.4).*

In particular, setting $\epsilon = \sqrt{\delta}$ gives a $(1-\sqrt{\delta})$-approximate solution.

Proof. Denote the projection of the optimal point x^* on the plane by $(m^*, s^*) = (\mu^T x^*, \tau^T x^*)$. We apply the linear oracle with respect to the slope $(-a)$ of the tangent to the level set L_{λ^*} at (m^*, s^*). The value of the linear objective at the optimum is $b^* = s^* + am^*$, which is the y-intercept of the tangent line. The linear oracle returns a δ-approximate solution, that is a solution on a parallel line with y-intercept $b \leq \delta b^*$. Suppose the original (nonlinear) objective value at the returned solution is lower-bounded by λ, that is it lies on a line tangent to L_λ (See Figure 3.5(b)). From Lemma 3.6.1, we have $b - b^* = s^*\left[1 - \left(\frac{\lambda}{\lambda^*}\right)^2\right]$, therefore

$$\left(\frac{\lambda}{\lambda^*}\right)^2 = 1 - \frac{b-b^*}{s^*} = 1 - \left(\frac{b-b^*}{b^*}\right)\frac{b^*}{s^*} \geq 1 - \delta\frac{b^*}{s^*}. \tag{3.7}$$

Recall that $b^* = s^* + m^*\frac{2\sqrt{s^*}}{\lambda^*}$ and $m^* \leq (1-\epsilon)t$, then

$$\frac{b^*}{s^*} = 1 + \frac{2m^*}{\lambda^*\sqrt{s^*}} = 1 + \frac{2m^*}{t-m^*} \leq 1 + \frac{2m^*}{\frac{\epsilon}{1-\epsilon}m^*} = 1 + \frac{2(1-\epsilon)}{\epsilon} = \frac{2-\epsilon}{\epsilon}.$$

Together with Eq. (3.7), this gives a $\sqrt{1-\delta\frac{2-\epsilon}{\epsilon}}$-approximation factor to the optimal.

On the other hand, setting $\epsilon = \sqrt{\delta}$ gives the approximation factor $\sqrt{1-\delta\frac{2-\sqrt{\delta}}{\sqrt{\delta}}} = 1 - \sqrt{\delta}$. □

Next, we prove a geometric lemma that will be needed to analyze the approximation factor we get when applying the linear oracle on an approximately optimal slope.

Lemma 3.6.3. *Consider the level set L_λ and points (m^*, s^*) and (m, s) on it, at which the tangents to L_λ have slopes $-a$ and $-a(1+\xi)$ respectively. Let the y-intercepts of the tangent line at (m, s) and the line parallel to it through (m^*, s^*) be b_1 and b respectively. Then $\frac{b}{b_1} \leq \frac{1}{1-\xi^2}$.*

Proof. The equation of the level set L_λ is $y = (\frac{t-x}{\lambda})^2$ so the slope at a point $(m, s) \in L_\lambda$ is given by the derivative at $x = m$, that is $-\frac{2(t-m)}{\lambda^2} = -\frac{2\sqrt{s}}{\lambda}$. So, the slope of the tangent to the level set L_λ at point (m^*, s^*) is $-a = -\frac{2\sqrt{s^*}}{\lambda}$. Similarly the slope of the tangent at (m, s) is $-a(1+\xi) = -\frac{2\sqrt{s}}{\lambda}$. Therefore, $\sqrt{s} = (1+\xi)\sqrt{s^*}$, or equivalently $(t-m) = (1+\xi)(t-m^*)$.

Since b, b_1 are intercepts with the y-axis, of the lines with slopes $-a(1+\xi) = -\frac{2\sqrt{s}}{\lambda}$ containing the points (m^*, s^*), (m, s) respectively, we have

$$\begin{aligned} b_1 &= s + \frac{2\sqrt{s}}{\lambda} m = \frac{t^2 - m^2}{\lambda^2} \\ b &= s^* + (1+\xi)\frac{2\sqrt{s^*}}{\lambda} m^* = \frac{t - m^*}{\lambda^2}(t + m^* + 2\xi m^*). \end{aligned}$$

Therefore

$$\begin{aligned} \frac{b}{b_1} &= \frac{(t-m^*)(t+m^*+2\xi m^*)}{(t-m)(t+m)} = \frac{1}{1+\xi}\frac{t+m^*+2\xi m^*}{t+m} = \frac{1}{1+\xi}\frac{t+(1+2\xi)m^*}{(1-\xi)t+(1+\xi)m^*} \\ &\leq \frac{1}{1+\xi}\left(\frac{1}{1-\xi}\right) = \frac{1}{1-\xi^2}, \end{aligned}$$

where we use $m = t - (1+\xi)(t - m^*)$ from above. □

We now show that we get a good approximation even when we use an approximately optimal weight vector with our oracle.

Lemma 3.6.4. *Suppose that we use an approximately optimal weight vector with a δ-approximate linear oracle (3.2) for solving the nonconvex threshold problem (3.4). In particular, suppose the weight vector (slope) that we use is within $(1+\xi)$ of the slope of the tangent at the optimal solution. Then this will give a solution to the nonconvex threshold problem (3.4) with value at least $\sqrt{1 - \left[\frac{\delta}{1-\xi^2} - 1\right]\frac{2-\epsilon}{\epsilon}}$ times the optimal, provided the optimal solution satisfies $\mu^T x^* \leq (1-\epsilon)t$.*

Proof. Suppose the optimal solution is (m^*, s^*) and it lies on the optimal level set λ^*. Let the slope of the tangent to the level set boundary at the optimal solution be $(-a)$. We apply our δ-approximation linear oracle with respect to slope that is $(1+\xi)$ times the optimum slope $(-a)$. Suppose the resulting black box solution lies on the line with y-intercept b_2, and the true optimum lies on the line with y-intercept b'. We know $b' \in [b_1, b]$, where b_1 and b are the y-intercepts of the lines with slope $-(1+\xi)a$ that are tangent to L_{λ^*} and pass through (m^*, s^*) respectively. Then we have $\frac{b_2}{b} \leq \frac{b_2}{b'} \leq \delta$.

Furthermore, by Lemma 3.6.3 we have $\frac{b}{b_1} \leq \frac{1}{1-\xi^2}$.

On the other hand, from Lemma 3.6.1, $b_2 - b_1 = s[1 - (\frac{\lambda_2}{\lambda^*})]$, where λ_2 is the smallest possible objective function value along the line with slope $-a(1+\xi)$ and y-intercept b_2, in other words the smallest possible objective function value that the solution returned by the approximate linear oracle may have; (m, s) is the tangent point of the line with slope $-(1+\xi)a$, tangent to L_{λ^*}.

Therefore, applying the above inequalities, we get

$$\left(\frac{\lambda_2}{\lambda^*}\right)^2 = 1 - \frac{b_2 - b_1}{s} = 1 - \frac{b_2 - b_1}{b_1}\frac{b_1}{s} = 1 - \left(\frac{b_2}{b}\frac{b}{b_1} - 1\right)\frac{b_1}{s} \geq 1 - \left(\frac{\delta}{1-\xi^2} - 1\right)\frac{2-\epsilon}{\epsilon},$$

where $\frac{b_1}{s} \leq \frac{2-\epsilon}{\epsilon}$ follows as in the proof of Lemma 3.6.2. The result follows. □

Consequently, we can approximate the optimal solution by applying the approximate linear oracle on a small number of appropriately chosen linear functions and picking the best resulting solution.

Theorem 3.6.5 (Theorem 3.4.3 in main body.). *Suppose we have a δ-approximation linear oracle for problem (3.2). Then, the nonconvex threshold problem (3.4) has a $\sqrt{1-\left[\frac{\delta-(1-\epsilon^2/4)}{(2+\epsilon)\epsilon/4}\right]}$ -approximation algorithm that calls the linear oracle a logarithmic in the input and polynomial in $\frac{1}{\epsilon}$ number of times, assuming the optimal solution to (3.4) satisfies $\mu^T\mathbf{x}^* \leq (1-\epsilon)t$.*

Proof. The algorithm applies the linear approximation oracle with respect to a small number of linear functions, and chooses the best resulting solution. In particular, suppose the optimal slope (tangent to the corresponding level set at the optimal solution point) lies in the interval $[L, U]$ (for lower and upper bound). We find approximate solutions with respect to the slopes $L, L(1+\xi), L(1+\xi)^2, ..., L(1+\xi)^k \geq U$, namely we apply the approximate linear oracle $\frac{\log(U/L)}{\log(1+\xi)}$ times, where $\xi = \frac{\epsilon^3}{2(1+\epsilon^3)}$. With this, we are certain that the optimal slope will lie in some interval $[L(1+\xi)^i, L(1+\xi)^{i+1}]$ and by Lemma 3.6.4 the solution returned by the linear oracle with respect to slope $L(1+\xi)^{i+1}$ will give a $\sqrt{1-\left[\frac{\delta}{1-\xi^2}-1\right]\frac{2-\epsilon}{\epsilon}}$- approximation to our non-linear objective function value. Since we are free to choose ξ, setting it to $\xi = \epsilon/2$ gives the desired number of queries.

We conclude the proof by noting that we can take L to be the slope tangent to the corresponding level set at (m_L, s_L) where s_L is the minimum positive coordinate of the variance vector and $m_L = t(1-\epsilon)$. Similarly let U be the slope tangent at (m_U, s_U) where $m_U = 0$ and s_U is the sum of coordinate of the variance vector. □

When $\delta = 1$, that is when we can solve the underlying linear problem exactly in polynomial time, the above algorithm gives an approximation factor of $\sqrt{\frac{1}{1+\epsilon/2}}$ or equivalently $1-\epsilon'$ where $\epsilon = 2[\frac{1}{(1-\epsilon')^2}-1]$. While the number of calls to the linear oracle that this algorithm makes is still logarithmic in the problem input and polynomial in $\frac{1}{\epsilon}$, the result is a bi-criteria approximation: It requires that there is a small gap between the mean of the optimal solution and t so it is weaker than our previous algorithm, which had no such requirement. This is expected, since of course this algorithm is cruder, taking a single geometric progression of linear functions rather than tailoring the linear oracle applications to the objective function value that it is searching for, as in the case of the nonlinear separation oracle that the previous algorithm from Section 3.5 is based on.

■ 3.7 Discussion

Other potential approaches We now include a brief discussion on where traditional other approaches fail in solving the nonconvex problems we consider.

Radzik gives a black-box solution for combinatorial optimization with rational objectives that are a ratio of two linear functions, by converting the rational objective into a linear constraint. A key property of the rational function that allows for an efficient algorithm is that it is *monotone* along the boundary of the feasible set; this is not the case for any of our objective functions and is one of the biggest challenges in working with nonconvex optimization problems: greedy, local search, interior point and other standard techniques do not work.

Our approach is conceptually very different from previous analyses of related problems. Common approximation techniques for hard instances of stochastic and multicriteria problems convert pseudopolynomial algorithms to FPTAS by scaling and rounding [128, 117], or they discretize the decision space and use a combination of dynamic programming and enumeration of possible feasible solutions over this cruder space [52]. In most cases the techniques are intimately intertwined with the structure of the underlying combinatorial problem and cannot extend to arbitrary problems. In contrast, the near-optimal efficiency of our algorithms is due to the fact that we carefully analyze the form of the objective function and use a "top-down" approach where our knowledge of the objective function level sets guides us to zoom down into the necessary portion of the feasible space.

Concluding remarks This chapter has presented an intuitive conceptual technique for approximating large nonconvex programs and derived from it efficient approximation schemes for a broad class of stochastic problems that incorporate risk. Our algorithms are independent of the feasible set structure and use solutions for the underlying linear (deterministic) problems as oracles for solving the nonconvex (stochastic) counterparts. As such they apply to very general combinatorial and discrete, as well as continuous settings, for which *exact* or *approximate* linear oracles are available.

We conjecture that our nonconvex approximation technique would yield practical polynomial-time approximation algorithms for quasi-concave minimization, quasi-convex maximization and other nonconvex functions of low rank (*i.e.*, that can be thought of as functions on a low-dimensional subspace), attaining their maxima at extreme points of the feasible set. Currently, the analysis of our algorithms is specific to the objective functions at hand. It is desirable to find a unifying general proof of the low complexity of the nonlinear oracles and categorize for which classes of functions such low complexity is possible. This motivates further research in geometry on polytopal approximation of convex and nonconvex bodies (given by the lower or upper level sets of the objective function), where the approximation is with respect to the function values as opposed to volume or surface as in existing work on polytopal approximation in convex geometry [60].

From a practical point of view, we have implemented our approximation algorithms and seen very promising empirical results: for example, finding a 99.99%-approximation of the optimal stochastic shortest path in grid networks of up to $40,000$ nodes with randomly generated mean and variance edge values takes only up to 6 oracle calls to a deterministic shortest path algorithm (See Section 6.4.4)!

In a different line of research, it would be interesting to extend our *offline nonlinear* to *online nonlinear*, or *online stochastic* models, as has previously been done with offline *linear* to online linear problems [72, 71].

CHAPTER 4

Low-rank nonconvex optimization

In this chapter we provide a different technique for approximating low-rank nonconvex optimization problems (more precisely, quasi-concave minimization or quasi-convex maximization), motivated by the nonconvex problems arising from stochastic optimization with risk. It applies to arbitrary functions with *bounded gradient* and *positive optimal value* over *polyhedral feasible sets*.

The algorithms are randomized and have expected polynomial running time. They enumerate the extreme points of a projection of the feasible set onto the low-dimensional subspace determined by the function and select the best extreme point. The extreme points of the projected feasible set contain the optimum by the property of quasi-concave functions [11]. Although the number of such extreme points is exponential in the worst case, it is polynomial on average and in a smoothed sense. This leads to an expected polynomial time approximation algorithm. Below, we first present the average and smoothed bounds on the number of extreme points, and then more formally describe and analyze the algorithms.

4.1 Rank 2

Consider projecting a polytope onto a subspace of dimension 2 spanned by vectors $\mathbf{u}, \mathbf{w} \in \mathbb{R}^m$. We show that if $\mathbf{u}, \mathbf{w}$ are uniformly random unit vectors or fixed vectors which are slightly perturbed, then the expected number of extreme points on the polytope projection onto $span(\mathbf{u}, \mathbf{w})$ is linear, which will yield algorithms for rank-2 quasi-concave minimization with a low expected polynomial running time. The techniques in this section are motivated by the recent techniques of Kelner and Spielman [79] for the polynomial simplex algorithm for linear programming (which is a special case of rank-1 quasi-concave minimization).

To simplify the exposition in this section, we focus on polytopes with 01-vertices, namely vertices which are a subset of the vertices of the unit hypercube. We remark that a polytope corresponding to the feasible set of a combinatorial optimization problem is typically of this form. Then:

Fact 4.1.1. *Each edge of the polytope P has length of at least* 1.

Fact 4.1.2. *The polytope P is contained in the unit hypercube, which in turn is contained in a ball with radius $\sqrt{m}/2$.*

Average bounds

Theorem 4.1.3. *Let* $\mathbf{u}, \mathbf{w} \in \mathbb{R}^m$ *be uniformly random unit vectors and let* V *be their span. Then the expectation of the number of edges on the projection of* P *onto* V *is at most* $2\sqrt{2}\pi m$*.*

Proof. By Fact 4.1.2, the perimeter of the projection (which we sometimes call *shadow*) of P onto V is bounded above by $\pi\sqrt{m}$. Next, for each edge I of the polytope P, denote by $S_I(V)$ the event that edge I appears in the shadow, and let $l(I)$ be the length of the edge in the shadow. The sum of expected edge lengths in the shadow is at most equal to the biggest possible perimeter:

$$\sum_I \mathbf{E}[l(I)] = \sum_I \mathbf{E}[l(I)|S_I(V)]\,\mathbf{Pr}[S_I(V)] \le \pi\sqrt{m}.$$

By Lemma 4.1.4 below, $\mathbf{E}[l(I)|S_I(V)] \ge \frac{1}{2\sqrt{2m}}$. Therefore,

$$\mathbf{E}[\text{number of shadow edges}] = \sum_I \mathbf{Pr}[S_I(V)] \le 2\sqrt{2}\pi m,$$

where m is the dimension of the polytope P, in our case it is the number of edges of the original graph. □

Lemma 4.1.4. *For all edges* I *of the polytope* P*,* $\mathbf{E}[l(I)|S_I(V)] \ge \frac{1}{2\sqrt{2m}}$*.*

Proof. We first note a direct corollary from Lemma 3.8 in Kelner & Spielman [79], namely that if an edge I of the polytope appears in the shadow, it will likely make a small angle $\theta_I(V)$ with the projection plane V, $\mathbf{Pr}_V\left[\cos(\theta_I(V)) \ge \frac{1}{\sqrt{2m}} \mid S_I(V)\right] \ge \frac{1}{2}$.

Now, since any edge in the polytope P has length at least 1 (by Fact 4.1.1 above), the length of the edge in the shadow would be at least $\cos(\theta_I(V))$ and its expectation provided it appears in the shadow is

$$\mathbf{E}[l(I)|S_I(V)] \ge \frac{1}{\sqrt{2m}}\frac{1}{2}.$$

□

Smoothed bounds

We now provide smoothed results for rank-2 quasi-concave minimization. In particular, we show that the expected number of extreme points (equivalently edges) on the projection of a general $0-1$ vertex polytope onto a perturbed plane is polynomial in m and $1/\rho$, the inverse of our perturbation.

We first define a ρ-perturbation of the vector $\mathbf{u}$, for $\rho > 0$.

Definition 4.1.5. *Choose an angle* $\theta \in [0, \pi]$ *at random from an exponential distribution with mean* ρ*, restricted to the range* $[0, \pi]$*. Set the* ρ*-perturbation of* $\mathbf{u}$ *to be a unit vector chosen uniformly at random at an angle* θ *to* $\mathbf{u}$*.*

The following theorem states that the expected number of edges on the polytope shadow is polynomial.

Theorem 4.1.6. *Let $\mathbf{u}_1, \mathbf{u}_2 \in \mathbb{R}^m$ be given vectors and let $\mathbf{v}_1$ and $\mathbf{v}_2$ be their respective ρ-perturbations. Denote $V = \operatorname{span}(\mathbf{v}_1, \mathbf{v}_2)$. The expected number of edges of the projection of P onto V is at most $4\pi\sqrt{2m}/\rho$, for $\rho < 1/\sqrt{m}$.*

The theorem follows similarly to the argument in Section 4.1 from the next lemma.

Lemma 4.1.7. *With the variables above, $\mathbf{Pr}_{\mathbf{v}_1,\mathbf{v}_2}[\cos(\theta_I(V)) \leq \epsilon \mid S_I(V)] \leq 4(\epsilon/\rho)^2$.*

The proof is very similar to that of Kelner and Spielman [79], but it must deal with a different distribution of 2-planes and the fact that P is not full-dimensional or simple, both of which introduce some new technical difficulties. We include it here for completeness.

Proof. Write the inequality constraints of the polytope P in the form $\mathbf{a}_i^T\mathbf{x} \leq 1$, and write the equality constraints as $\mathbf{b}_i^T\mathbf{x} = t_i$ for some constants t_i. Let $\mathbf{a}_1, \dots, \mathbf{a}_k$ and $\mathbf{b}_1, \dots, \mathbf{b}_\ell$ be the constraints that are tight on I, so that I lies on the line

$$\{\mathbf{x} \mid \mathbf{a}_i^T\mathbf{x} = 1,\ \mathbf{b}_i^T\mathbf{x} = t_i, \quad i = 1, \dots, k,\ j = 1, \dots, \ell\}.$$

Furthermore, let $W = \operatorname{span}(\mathbf{a}_1, \dots, \mathbf{a}_k, \mathbf{b}_1, \dots, \mathbf{b}_\ell)$, let the cone $C = \{\sum_{i=1}^k \alpha_i\mathbf{a}_i + \sum_{j=1}^\ell \beta_k\mathbf{b}_j \mid \alpha_i \geq 0,\ \beta_j \in \mathbb{R}\}$, and let $\mathbf{q}$ be a unit vector that is parallel to the edge I. Note that W is the subspace of $\mathbb{R}^m$ that is orthogonal to $\mathbf{q}$.

A standard result from convex geometry [15] says that $S_I(V)$ will hold if and only if V and C intersect nontrivially. For notational simplicity, we shall denote this with $V \cap C \neq \emptyset$, although obviously this intersection always includes the origin. We observe that $C \subseteq W$, so $V \cap C \subseteq V \cap W$.

Rewrite $\mathbf{v}_1$ and $\mathbf{v}_2$ in a new coordinate system that will enable us to evaluate $\theta_I(V)$ and to determine whether $S_I(V)$ holds. We do so as follows (see Figure 4.1):

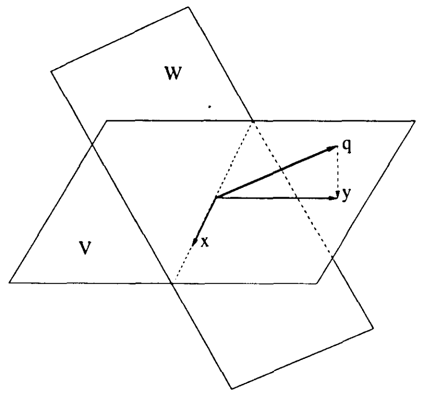


Figure 4.1. Some of the variables in our change of coordinates. Picture taken from [79].

1. Let $\mathbf{y}$ be the unit vector through the projection of $\mathbf{q}$ onto V. Note that $\theta_I(V)$ equals the angle between $\mathbf{q}$ and $\mathbf{y}$.

2. Fix some arbitrary $\mathbf{c} \in C$, and let $\mathbf{x}$ be the unique vector in V that is orthogonal to $\mathbf{y}$ and has positive inner product with $\mathbf{c}$. Note that $\mathbf{x}$ is orthogonal to $\mathbf{q}$, so $\mathbf{x} \in V \cap W$. By the discussion above, this means that $S_I(V)$ holds if and only if $\mathbf{x} \in C$.

3. We now have unit vectors $\mathbf{x}$ and $\mathbf{y}$ such that $\operatorname{span}(\mathbf{x}, \mathbf{y}) = \operatorname{span}(\mathbf{v}_1, \mathbf{v}_2)$. Define the angles α and β by

$$\begin{aligned}\mathbf{v}_1 =& \mathbf{x}\cos\alpha + \mathbf{y}\sin\alpha, \text{ and}\\ \mathbf{v}_2 =& \mathbf{x}\cos\beta + \mathbf{y}\sin\beta.\end{aligned}$$

4. Let $\theta = \theta_I(V)$ be the angle between $\mathbf{q}$ and $\mathbf{y}$. Once this is specified, $\mathbf{y}$ is constrained to lie on an $(m-2)$-dimensional sphere. Let $\mathbf{z}$ be the corresponding point on this sphere.

Some of the new variables are shown in Figure 4.1. Intuitively, we started out by expressing V as the span of two arbitrary vectors, $\mathbf{v}_1$ and $\mathbf{v}_2$. We changed variables to instead write V in terms of its intersection with W (which is the variable $\mathbf{x}$) and the orthogonal complement of this intersection (which is the variable $\mathbf{y}$). The variables $\mathbf{x}$ and $\mathbf{y}$ specify the plane spanned by $\mathbf{v}_1$ and $\mathbf{v}_2$, but they don't specify where in the plane these variables lie. We thus introduced two new variables, α and β, to replace these lost degrees of freedom.

As computed by Deshpande and Spielman [30], the Jacobian of the change of variables from $\mathbf{v}_1$ and $\mathbf{v}_2$ to $\mathbf{x}$, $\mathbf{z}$, θ, α, and β is given by

$$d\mathbf{v}_1\, d\mathbf{v}_2 = c\cos(\theta)\sin(\theta)^{m-3}\sin(\alpha-\beta)^{m-2} d\mathbf{x}\, d\mathbf{z}\, d\theta\, d\alpha\, d\beta,$$

where c is a constant depending only on m.

Let η_1 and η_2 denote the probability density functions of $\mathbf{v}_1$ and $\mathbf{v}_2$ respectively. The probability that we aim to bound is given by

$$\begin{aligned}\mathbf{Pr}[\cos(\theta_I(V) \leq \epsilon \mid S_I(V)] &= \frac{\int_{\mathbf{v}_1, \mathbf{v}_2 \in S^{m-1},\, V\cap C \neq \emptyset,\, \theta_I(V) \leq \epsilon} \eta_1(\mathbf{v}_1)\eta_2(\mathbf{v}_2) d\mathbf{v}_1\, d\mathbf{v}_2}{\int_{\mathbf{v}_1, \mathbf{v}_2 \in S^{m-1},\, V\cap C \neq \emptyset} \eta_1(\mathbf{v}_1)\eta_2(\mathbf{v}_2) d\mathbf{v}_1\, d\mathbf{v}_2}\\ &= \frac{\int_{\theta > \cos^{-1}(\epsilon),\, \mathbf{x}\in C,\, \mathbf{z},\, \alpha,\, \beta} c\cos(\theta)\sin(\theta)^{m-3}\sin(\alpha-\beta)^{m-2}\eta_1(\mathbf{v}_1)\eta_2(\mathbf{v}_2) d\mathbf{x}\, d\mathbf{z}\, d\theta\, d\alpha\, d\beta}{\int_{\mathbf{x}\in C,\, \theta,\, \mathbf{z},\, \alpha,\, \beta} c\cos(\theta)\sin(\theta)^{m-3}\sin(\alpha-\beta)^{m-2}\eta_1(\mathbf{v}_1)\eta_2(\mathbf{v}_2) d\mathbf{x}\, d\mathbf{z}\, d\theta\, d\alpha\, d\beta},\end{aligned}$$

Suppose that we fix $\mathbf{x}$, $\mathbf{z}$, α, and β. The only variable that's left is θ, so we can write $\mathbf{v}_1$ and $\mathbf{v}_2$ as functions of θ, and we can write the probability density function as

$$\eta(\theta) = \eta_1(\mathbf{v}_1(\theta))\eta_2(\mathbf{v}_2(\theta)).$$

As we vary θ, we can bound how rapidly $\eta(\theta)$ can change. In particular, if we change θ by ϕ, we note that both $\mathbf{v}_1$ and $\mathbf{v}_2$ rotate by an angle of at most ϕ. It follows that for all $\phi < \rho$ and all θ,

$$\eta_1(\mathbf{v}_1(\theta)) < \frac{\eta_1(\mathbf{v}_1(\theta+\phi))}{e}$$

and

$$\eta_2(\mathbf{v}_2(\theta)) < \frac{\eta_2(\mathbf{v}_2(\theta+\phi))}{e}$$

so

$$\eta(\theta) < \frac{\eta(\theta+\phi)}{e^2}.$$

This allows us to bound our integral as

$$\frac{\int_{\theta > \cos^{-1}(\epsilon),\, \mathbf{x}\in C,\, \mathbf{z},\, \alpha,\, \beta} c\cos(\theta)\sin(\theta)^{m-3}\sin(\alpha-\beta)^{m-2}\eta(\theta)d\mathbf{x}\,d\mathbf{z}\,d\theta\,d\alpha\,d\beta}{\int_{\mathbf{x}\in C,\, \theta,\, \mathbf{z},\, \alpha,\, \beta} c\cos(\theta)\sin(\theta)^{m-3}\sin(\alpha-\beta)^{m-2}\eta(\theta)d\mathbf{x}\,d\mathbf{z}\,d\theta\,d\alpha\,d\beta}$$

$$\le \max_{\mathbf{x},\mathbf{z},\alpha\beta} \frac{\int_{\theta=\cos^{-1}(\epsilon)}^{\pi/2} c\cos(\theta)\sin(\theta)^{m-3}\sin(\alpha-\beta)^{m-2}\eta(\theta)d\theta}{\int_{\theta=0}^{\pi/2} c\cos(\theta)\sin(\theta)^{m-3}\sin(\alpha-\beta)^{m-2}\eta(\theta)d\theta}$$

$$\le \frac{e^2\int_{\theta=\cos^{-1}(\epsilon)}^{\pi/2} c\cos(\theta)\sin(\theta)^{m-3}\sin(\alpha-\beta)^{m-2}\eta(\theta)d\theta}{\int_{\theta=0}^{\pi/2} c\cos(\theta)\sin(\theta)^{m-3}\sin(\alpha-\beta)^{m-2}\eta(\theta)d\theta}$$

$$= e^2\frac{(\sin\theta)^{m-2}\big|_{\cos^{-1}(\epsilon)}^{\pi/2}}{(\sin\theta)^{m-2}\big|_{\pi/2-\rho}^{\pi/2}}$$

$$= e^2\frac{1-(\sin(\cos^{-1}(\epsilon)))^{m-2}}{1-(\sin(\rho))^{m-2}}$$

$$\le e^2\frac{1-(1-\epsilon^2)^{(m-2)/2}}{1-(1-\rho^2/2)^{m-2}}$$

$$\le e^2\frac{(m-2)\epsilon^2}{(m-2)4(1-1/\sqrt{e})\rho^2},\ \text{as}\ \rho < 1/\sqrt{m},$$

$$\le 4(\epsilon/\rho)^2.$$

□

Naturally, the smaller the perturbation, the weaker the bound in the theorem. However, setting $\rho = \frac{1}{\sqrt{2m}}$, for example, gives the linear bound $8\pi m$ which is just a little larger than the bound on the number of shadow edges for the average case. Finally, note that by Lemma 6.1.3, these bounds imply linear (in the number of graph edges) average and smoothed bounds for the number of optimal paths in the parametric shortest paths problem as well.

4.2 Rank k: Definitions

Low-Rank Quasi-Concave Minimization In this section, we define the rank of a function and the shadow of the feasible set onto a subspace, and we state some properties of the global minima of low-rank quasi-concave functions.

Definition 4.2.1. *The* rank k *of function* $f : \mathbb{R}^n \to \mathbb{R}$ *is the smallest integer for which the function can be written in the form* $f(\mathbf{x}) = \hat{f}(\mathbf{a}_1^T\mathbf{x}, ..., \mathbf{a}_k^T\mathbf{x})$ *for some function* $\hat{f} : \mathbb{R}^k \to \mathbb{R}$ *and linearly independent vectors* $\mathbf{a}_1, ..., \mathbf{a}_k \in \mathbb{R}^n$. [1]

Of course, every function on $\mathbb{R}^n$ can be trivially written as a rank-n function $f(x_1, ..., x_n) = f(\mathbf{e}_1^T\mathbf{x}, ..., \mathbf{e}_n^T\mathbf{x})$ where $\mathbf{e}_i$ has 1 in its ith coordinate and 0 everywhere else. Low-rank functions are

[1]Our definition of rank may differ by 1 from other definitions that have appeared in the literature [112].

ones whose rank k is much smaller than n. We call the subspace spanned by vectors $a_1, ..., a_k$ the *rank* or *projection subspace* corresponding to the function f. Further, we can project the feasible set in $\mathbb{R}^n$ onto the rank subspace. We will call this projection the *(k-dimensional) shadow*. The following claim is readily verified:

Claim 4.2.2. *The minimum of $\hat{f}$ over the shadow of the polytope Q in the rank subspace $\mathbb{R}^k$ is the same as the minimum of f over Q in $\mathbb{R}^n$, and it is attained at an extreme point of the shadow.*

If the feasible polytope Q is given as a polynomial set of constraints and the quasi-concave function f is specified by oracle queries, we can find the minimum by enumerating the vertices of the shadow (and keeping track of their corresponding vertices in Q). This can be done, for example, by a depth-first search exploration of the associated shadow graph, where we can find all neighbors of a vertex in the shadow by checking all neighbors of the corresponding vertex in the original polytope.

We can use similar enumeration approaches for more general models of specification of the objective function and the feasible polytope; this is not the focus of our work, so we limit our discussion here.

We next describe our model for perturbing low-rank functions.

Sampling from the Grassmannian

To properly handle our probability calculations, we shall need to define precisely the spaces and distributions from which we are sampling.

Definition 4.2.3. *The* Grassmannian $\mathrm{Gr}(n, k)$ *is the set of k-dimensional subspaces of $\mathbb{R}^n$.*

Rather than sampling from the Grassmannian directly, we shall draw our samples from the space $O(n)$ of orthogonal $n \times n$ matrices and take the subspace spanned by the first k columns.

Definition 4.2.4. *Let $\Pi_k : \mathbb{R}^{n^2} \to \mathbb{R}^{nk}$ be the map that takes an $n \times n$ matrix to its first k columns, and let $\overline{\Pi}_k : \mathbb{R}^{n^2} \to \mathrm{Gr}(n, k)$ be the map that takes a matrix to the subspace spanned by its first k columns.*

The space of orthogonal matrices has a standard measure, known as the *Haar measure* [24], that properly encapsulates what we mean by a "uniformly random" orthogonal matrix. It is the measure induced by treating the space of orthogonal matrices as a subset of $\mathbb{R}^{n^2}$, and it is the unique rotation-invariant measure on $O(n)$. The Haar measure and the map $\overline{\Pi}_k$ then induce a measure on $\mathrm{Gr}(n, k)$. When we take integrals over $O(n)$ or $\mathrm{Gr}(n, k)$, it shall be with respect to these measures.

Perturbing a low-rank function

We defined a function $f : \mathbb{R}^n \to \mathbb{R}$ to have rank k if it can be written in the form $f(\mathrm{x}) = \hat{f}(\mathrm{a}_1^T\mathrm{x}, \mathrm{a}_2^T\mathrm{x}, ..., \mathrm{a}_k^T\mathrm{x})$ for some function $\hat{f} : \mathbb{R}^k \to \mathbb{R}$ and linearly independent vectors $\mathrm{a}_1, ..., \mathrm{a}_k \in \mathbb{R}^n$. Equivalently, we can define a rank-k function as a function $\hat{f} : \mathbb{R}^k \to \mathbb{R}$ and a k-dimensional subspace S.

Consequently, we can describe a perturbation in two equivalent ways, as a perturbation of the subspace or of the function. For notational simplicity, we shall choose the former option, although our results could easily be restated in terms of the latter.

In order to define a perturbation of a subspace, we shall need a notion of the distance between two matrices, which we take to be the spectral norm of their difference:

$$\operatorname{dist}(Q,T) = \max_{\|\mathbf{x}\|=1}(Q-T)\mathbf{x},$$

where $\|\cdot\|$ is the L_2 norm.

As mentioned above, we shall sample from the space of orthogonal matrices and then obtain elements of the Grassmannian $\operatorname{Gr}(n,k)$ by keeping only the first k columns. Therefore it will suffice for us to define a ρ-perturbation of an orthogonal matrix, as this will induce a similar notion on the Grassmannian, and thus on the space of low-rank quasi-concave functions.

Definition 4.2.5. *A ρ-perturbation of an orthogonal matrix Q_0 is an orthogonal matrix sampled from the distribution on orthogonal matrices whose density at a matrix Q is proportional to*

$$\exp\{\operatorname{dist}(Q^{-1}Q_0, \mathrm{Id})/\rho\}.$$

4.3 Rank k: Analysis

Our main theorem states that there is an expected polynomial-time smoothed algorithm for low-rank quasi-concave minimization over integral polytopes:

Theorem 4.3.1. *Let $P \subseteq \mathbb{R}^n$ be a polytope with integer vertices whose coordinates are all less than some constant γ and a polynomial number of facets, and let $f : P \longrightarrow \mathbb{R}$ be a quasi-concave function of constant rank k given as an oracle. There exists an algorithm that minimizes a ρ-perturbation of f in expected time*

$$\operatorname{poly}(n, \gamma, 1/\rho),$$

where the degree of the polynomial depends on k.

By the enumeration techniques described in Section 4.2, this will follow immediately from our main technical lemma, which is a bound on the expected number of extreme points in the projection of the polytope onto the rank subspace of the objective function; the bound is polynomial in the perturbation size ρ and the original dimension n, but exponential in k.

Lemma 4.3.2. *Let $P \subseteq \mathbb{R}^n$ be a polytope with integer vertices whose coordinates are all less than some constant γ (and with* no *restriction on the number of facets), and let $Q_0 \subseteq \mathbb{R}^n$ be a k-dimensional vector subspace. The expected number of vertices on the projection of P onto a ρ-perturbation of Q_0 has at most*

$$\operatorname{poly}(n, \gamma, 1/\rho)$$

vertices, where the degree of the polynomial depends on k.

In the remainder of this section, we shall prove our main technical lemma, Lemma 4.3.2. Our general method of proof is motivated by the techniques used by Kelner and Spielman to construct their polynomial-time simplex method for linear programming. However, while the overall approach of bounding the combinatorial complexity through volumetric arguments is the same, studying the projection onto a subspace of dimension greater than two inheres significant new technical difficulties. Resolving these requires us to analyze how the geometry of the Grassmannian or the orthogonal group changes under the change of coordinates induced by a fairly intricate matrix decomposition.

As it will simplify our notation slightly, we shall prove the lemma for the projection of a polytope onto a $(k+1)$-dimensional subspace instead of onto a k-dimensional one.

At a high level, our proof is quite simple. We shall begin by showing that every face of P has a fairly large volume. We shall then show that this remains true about the projection of P onto a ρ-perturbation of a $(k+1)$-dimensional subspace. As we have assumed that all of P's coordinates are bounded above by some constant γ, the surface area of the projection will be bounded above by some constant depending on k and γ. Since we will have a lower bound on the volume of each facet of the projection, this will imply an upper bound on the total number of facets, and thus on the total number of vertices, as desired.

4.3.1 The volume of faces of P

We begin by providing a lower bound on the volume of any face of P.

Lemma 4.3.3. *Let $P \subset \mathbb{R}^n$ be a polytope with integer vertices. The (k-dimensional) volume of every k-dimensional face of P is at least $1/k!$.*

Proof. Let F be the face in question, and suppose that F is the convex hull of the vertices $\mathbf{v}_1, \dots, \mathbf{v}_s \in \mathbb{R}^n$. Since F is k-dimensional, there exists some set of of $(k+1)$ of these vertices that does not lie in any $(k-1)$-dimensional affine subspace. Without loss of generality, let this set consist of $\mathbf{v}_1, \dots, \mathbf{v}_{k+1}$. Since $\operatorname{conv}(\mathbf{v}_1, \dots, \mathbf{v}_{k+1}) \subseteq \operatorname{conv}(\mathbf{v}_1, \dots, \mathbf{v}_s)$, it suffices to show that $\mathrm{Vol_k}(\operatorname{conv}(\mathbf{v}_1, \dots, \mathbf{v}_{k+1})) \geq 1/k!$.

Therefore we aim to bound the volume of a k-dimensional simplex Σ with integer vertices. By translating the entire simplex, if necessary, we can also assume that v_{k+1} is the origin. If V is the $n \times k$ matrix whose i^{th} column comprises the coordinates of v_i, we have that

$$\mathrm{Vol_k}(\Sigma) = \frac{1}{k!}\sqrt{\det(V^T V)}.$$

We have assumed that Σ has nonzero volume, so $\det(V^T V) \neq 0$. Since $V^T V$ is an integer matrix, its determinant is an integer. Therefore its absolute value is at least 1, so $\mathrm{Vol}_k(\Sigma) \geq 1/k!$, as desired. □

4.3.2 The volume of faces of the projection

We now aim to show that the projection of a given k-dimensional face F of an n-dimensional polytope P onto a perturbed $(k+1)$-dimensional subspace has a sufficiently large volume, contingent upon F appearing on the boundary of the projection.

The proof of this result is rather technical. In order to provide some geometric intuition for why it should be true, we shall begin by considering some simpler variants of the problem. In the first such variant, we shall replace our perturbed subspace by a uniformly random one, and we shall remove the conditioning constraint that F appears on the boundary of the projection. We shall then add our conditioning constraint back in but still consider a uniformly random subspace. After that, we shall progress to our final, most general version.

Projecting onto a uniform subspace with no conditioning

We begin with the following problem: we are given some k-dimensional face F embedded in $\mathbb{R}^n$, and we choose a uniformly random $(k+1)$-dimensional subspace S. The goal is to show that the ratio

$$\frac{\mathrm{Vol}_k(\pi_S(F))}{\mathrm{Vol}_k(F)}$$

is unlikely to be less than some inverse polynomial bound where π_S denotes orthogonal projection onto S.

In this case, all that matters about F is the k-dimensional affine subspace T that it spans, and we are just trying to provide lower bounds on the determinant of the projection map from T onto S. When k equals 1, this amounts to understanding the length of the projection of a uniformly random unit vector onto some fixed unit vector, which is a very well-analyzed question. In this case, the expected length grows asymptotically like $1/\sqrt{n}$. In the higher dimensional case at hand, this is a well-studied question in random matrix theory (see Mehta [90]) whose answer is given by

$$\frac{\Gamma\left(\frac{d+1}{2}\right)\Gamma\left(\frac{n-d+1}{2}\right)}{\sqrt{\pi}\Gamma\left(\frac{n+1}{2}\right)} \approx C(d)\frac{1}{n^{d/2}}.$$

Projecting onto a uniform subspace with conditioning

When we project our polytope onto a lower-dimensional subspace, not all of the faces appear on the boundary of the projection. In order to analyze the expected volumes of the faces that do appear, we must therefore condition our probability estimates on the event that a given face appears on the boundary of the projection. It is here that we must deviate from previous smoothed analyses and introduce more involved methods from geometry and random matrix theory.

Let $P \subseteq \mathbb{R}^n$ be the set of points that satisfy a collection of m linear constraints,

$$P = \{\mathbf{x} \mid \mathbf{a}_i^T\mathbf{x} \leq 1,\ i = 1, \ldots, m\}.$$

For a k-dimensional face F of P and a $(k+1)$-dimensional subspace S, let $A_F(S)$ be the event that F appears as a facet of the projection of P onto S. The following theorem provides a geometric criterion for when $A_F(S)$ occurs:

Theorem 4.3.4. *Let $\mathbf{a}_{i_1}, \ldots, \mathbf{a}_{i_s}$ be the constraints that are satisfied with equality at all points of F, and let $C = \mathrm{pos}(\mathbf{a}_{i_1}, \ldots, \mathbf{a}_{i_k})$ be their positive hull. The event $A_F(S)$ occurs if and only if the intersection $C \cap S \neq \emptyset$.*

Proof. This follows from standard convex geometry [15]. □

We note that for a generic polytope, there will be $s = n - k$ constraints that are tight at a k-dimensional face, so the cone C will be $(n-k)$-dimensional. The $(n-k)$-dimensional subspace spanned by C will thus generically intersect the random $(k+1)$-dimensional subspace S in a 1-dimensional ray, and the question is whether this ray lies in C. This will occur with a positive probability that is proportional to the k-dimensional volume of the intersection of C with the unit sphere. In order to condition on $A_F(S)$, we must therefore only consider subspaces S that intersect C. In all that follows, we shall assume that $n > 2k+2$. Since we are assuming k is a constant and studying the asymptotics as n gets large, this inheres no loss of generality.

Our goal is to bound the probability

$$\mathbf{Pr}_{S\in \mathrm{Gr}(n,k+1)}\left[\frac{\mathrm{Vol}_k(\pi_S(F))}{\mathrm{Vol}_k(F)} \le \epsilon \,|\, A_F(S)\right].$$

By rotating our coordinate system if necessary, we can take F parallel to the subspace $\mathrm{span}(\mathbf{e}_1,\dots,\mathbf{e}_k)$, where the $\mathbf{e}_i$ are our unit basis vectors. If V is an $n\times(k+1)$ matrix whose columns form an orthonormal basis for S, and $\mathcal{F} = \begin{bmatrix} \mathrm{Id}_k \\ \mathbf{0}_{n-k}\end{bmatrix}$, then

$$\frac{\mathrm{Vol}_k(\pi_S(F))}{\mathrm{Vol}_k(F)} = \sqrt{\det\left((\mathcal{F}^T V)(\mathcal{F}^T V)^T\right)},$$

and therefore

$$\mathbf{Pr}_{S\in \mathrm{Gr}(n,k+1)}\left[\frac{\mathrm{Vol}_k(\pi_S(F))}{\mathrm{Vol}_k(F)} \le \epsilon \,|\, A_F(S)\right] \tag{4.1}$$

$$= \frac{\displaystyle\int_{\substack{Q\in O(n),\, \overline{\Pi}_{k+1}(Q)\cap C\neq\emptyset \\ \sqrt{\det((\mathcal{F}^T\Pi_{k+1}(Q))(\mathcal{F}^T\Pi_{k+1}(Q))^T)}\le\epsilon}} 1\, dQ}{\displaystyle\int_{Q\in O(n),\, \overline{\Pi}_{k+1}(Q)\cap C\neq\emptyset,} 1\, dQ},$$

where C is defined as in the statement of Theorem 4.3.4, and dQ denotes the Haar measure on $O(n)$.

Right now, Q is being represented as an $n\times n$ matrix. In this representation, it is rather difficult to describe which matrices meet the other conditions of integration, which makes it difficult to evaluate the integrals. To remedy this, we shall change to a new coordinate system on $O(n)$ that is more conducive to this task. To do so, we shall make use of a matrix decomposition known as the *generalized cosine-sine (CS) decomposition.*

Theorem-Definition 4.3.5 ([54]). *Let*

$$Q = \left(\begin{array}{ccc} \text{sizes} & k & n-k \\ k+1 & Q_{11} & Q_{12} \\ n-k-1 & Q_{21} & Q_{22} \end{array}\right)$$

be an $n \times n$ orthogonal matrix. For any $k \leq n/2$, there exists a unique decomposition:

$$Q = \left(\begin{array}{c|c|c} \text{sizes} & k+1 & n-k-1 \\ k+1 & U & 0 \\ \hline n-k-1 & 0 & V \end{array}\right)$$

$$\times \left(\begin{array}{c|c|cc|c} \text{sizes} & k & 1 & k & n-2k-1 \\ k & C & 0 & -S & 0 \\ 1 & 0 & 1 & 0 & 0 \\ \hline k & S & 0 & C & 0 \\ n-2k-1 & 0 & 0 & 0 & \mathrm{Id}_{n-2k-1} \end{array}\right)$$

$$\times \left(\begin{array}{c|c|c} \text{sizes} & k & n-k \\ k & W^T & 0 \\ \hline n-k & 0 & Z^T \end{array}\right)$$

where

- *U, V, W, and Z are orthogonal matrices,*
- *S and C are positive diagonal matrices,*
- *the diagonal elements $c_1, \ldots, c_p$ of C are nondecreasing, and*
- $C^2 + S^2 = \mathrm{Id}_k$.

We call this decomposition the generalized CS decomposition of Q. *The angles $\theta_1, \ldots, \theta_k$ such that* $\cos(\theta_i) = c_i$ *are called the* principal angles *between the subspace* $\Pi_{k+1}(Q)$ *and the subspace* $\Pi_k(\mathcal{F})$.

Let

$$\lambda_i = \cos^2(\theta_i).$$

We shall change coordinates so that our variables comprise $\lambda_1, \ldots, \lambda_k$ along with the orthogonal matrices U, V, W, and Z.

Theorem 4.3.6. *The Jacobian of the change of variables described above is given by:*

$$dQ = \prod_{i<j}(\lambda_j - \lambda_i) \prod_{i=1}^{k}(1-\lambda_i)^{(n-2k-2)/2}$$
$$\times\, d\lambda_1 \ldots d\lambda_p\, dU\, dV\, dW\, dZ,$$

where dU, dV, dW, and dZ are the Haar measures on the appropriate-dimensional spaces of orthogonal matrices.

It is not difficult to see that

$$\det\left(\left(\mathcal{F}^T\Pi_{k+1}(Q)\right)\left(\mathcal{F}^T\Pi_{k+1}(Q)\right)^T\right) = \prod_{i=1}^{k}\lambda_i.$$

This allows us to re-express the right-hand side of equation (4.1) as

$$\frac{\displaystyle\int_{\substack{\Pi_1(Z)\in C,\\ \prod_i \lambda_i \le \epsilon^2}} \prod_{i<j}(\lambda_j-\lambda_i)\prod_{i=1}^{k}(1-\lambda_i)^{(n-2k-2)/2} \times d\lambda_1\ldots d\lambda_p\, dU dV dW dZ}{\displaystyle\int_{\Pi_1(Z)\in C}\prod_{i<j}(\lambda_j-\lambda_i)\prod_{i=1}^{k}\lambda_i(1-\lambda_i)^{(n-2k-2)/2} \times d\lambda_1\ldots d\lambda_p\, dU dV dW dZ}$$

$$= \frac{\displaystyle\int_{\prod_i \lambda_i\le\epsilon^2}\prod_{i<j}(\lambda_j-\lambda_i)\prod_{i=1}^{k}(1-\lambda_i)^{(n-2k-2)/2} \times d\lambda_1\ldots d\lambda_p}{\displaystyle\int\prod_{i<j}(\lambda_j-\lambda_i)\prod_{i=1}^{k}(1-\lambda_i)^{(n-2k-2)/2} \times d\lambda_1\ldots d\lambda_p.}$$

Let $\zeta_i = 1-\lambda_i$ and write $\boldsymbol{\zeta} = (\zeta_1,\ldots,\zeta_k)$ and $\boldsymbol{\zeta}' = (\zeta_1,\ldots,\zeta_{k-1})$ for brevity. By expanding the products in the numerator and denominator and collecting terms, we can rewrite the last expression in the string of equations above as

$$\frac{\displaystyle\int_{\prod_i(1-\zeta_i)\le\epsilon^2}\sum_{t\ge0}\zeta_k^{t+(n-2k-2)/2}p_t(\boldsymbol{\zeta}')d\boldsymbol{\zeta}}{\displaystyle\int\sum_{t\ge0}\zeta_k^{t+(n-2k-2)/2}p_t(\boldsymbol{\zeta}')d\boldsymbol{\zeta}}$$

for some multivariate functions $p_t(\zeta')$. Interchanging the summation with the integration yields:

$$\begin{aligned}
&\frac{\sum_{t\geq 0}\int_{\prod_i(1-\zeta_i)\leq\epsilon^2}\zeta_k^{t+(n-2k-2)/2}p_t(\zeta')d\zeta}{\sum_{t\geq 0}\int\zeta_k^{t+(n-2k-2)/2}p_t(\zeta')d\zeta}\\
&\leq \max_{t\geq 0}\frac{\int_{\prod_i(1-\zeta_i)\leq\epsilon^2}\zeta_k^{t+(n-2k-2)/2}p_t(\zeta')d\zeta}{\int\zeta_k^{t+(n-2k-2)/2}p_t(\zeta')d\zeta}\\
&\leq \max_{t\geq 0}\frac{\int_{(1-\zeta_k)\leq\epsilon^{2/k}}\zeta_k^{t+(n-2k-2)/2}p_t(\zeta')d\zeta}{\int\zeta_k^{t+(n-2k-2)/2}p_t(\zeta')d\zeta}\\
&= \max_{t\geq 0}\frac{\int_{\zeta_k=1-\epsilon^{2/k}}^{1}\zeta_k^{t+(n-2k-2)/2}d\zeta_k}{\int_{\zeta_k=0}^{1}\zeta_k^{t+(n-2k-2)/2}d\zeta_k}\\
&\leq 1-(1-\epsilon^{2/k})^{(n-2k)/2}\\
&\leq \frac{n-2k}{2}\epsilon^{2/k},
\end{aligned}$$

where the maxima are taken over all t for which $p_t \neq 0$.

4.3.3 The general case

We now consider the fully general case described in Lemma 4.3.2, where we examine the projection of P onto a ρ-perturbation of an arbitrary $(k+1)$-dimensional subspace, and we take $\rho < 1/(n-2k)$. This results in a nonuniform probability density over the space of orthogonal matrices from which we draw Q; let μ be this probability density. We aim to compute

$$\begin{aligned}
&\mathbf{Pr}_{S\in\mathrm{Gr}(n,k+1)}\left[\frac{\mathrm{Vol}_k(\pi_S(F))}{\mathrm{Vol}_k(F)}\leq\epsilon \mid A_F(S)\right]\\
&= \frac{\displaystyle\int_{\substack{Q\in O(n),\,\overline{\Pi}_{k+1}(Q)\cap C\neq\emptyset,\\ \sqrt{\det((\mathcal{F}^T\Pi_{k+1}(Q))(\mathcal{F}^T\Pi_{k+1}(Q))^T)}\leq\epsilon}}\mu(Q)\,dQ}{\displaystyle\int_{Q\in O(n),\,\overline{\Pi}_{k+1}(Q)\cap C\neq\emptyset}\mu(Q)\,dQ}\\
&= \frac{\displaystyle\int_{\substack{\Pi_1(Z)\in C,\\ \prod_i\lambda_i\leq\epsilon^2}}\prod_{i<j}(\lambda_j-\lambda_i)\prod_{i=1}^{k}(1-\lambda_i)^{(n-2k-2)/2}\times\mu(\lambda_1,\ldots,\lambda_k,U,V,W,Z)\times d\lambda_1\ldots d\lambda_p\,dU dV dW dZ}{\displaystyle\int_{\Pi_1(Z)\in C}\prod_{i<j}(\lambda_j-\lambda_i)\prod_{i=1}^{k}\lambda_i(1-\lambda_i)^{(n-2k-2)/2}\times\mu(\lambda_1,\ldots,\lambda_k,U,V,W,Z)\times d\lambda_1\ldots d\lambda_p\,dU dV dW dZ}.
\end{aligned}$$

We shall fix U, V, W, and Z arbitrarily and bound the resulting ratio of integrals over the λ_i. Once we fix U, V, W, and Z, our density depends only on the λ_i, and our expression becomes (in the notation of the previous section):

$$\frac{\displaystyle\int_{\prod_i \lambda_i \leq \epsilon^2} \prod_{i<j}(\lambda_j - \lambda_i) \prod_{i=1}^{k} (1-\lambda_i)^{(n-2k-2)/2} \times \mu(\lambda_1, \ldots, \lambda_k)\, d\lambda_1 \ldots d\lambda_p}{\displaystyle\int \prod_{i<j}(\lambda_j - \lambda_i) \prod_{i=1}^{k} \lambda_i (1-\lambda_i)^{(n-2k-2)/2} \times \mu(\lambda_1, \ldots, \lambda_k)\, d\lambda_1 \ldots d\lambda_p}$$
$$= \frac{\int_{\prod_i (1-\zeta_i) \leq \epsilon^2} \sum_{t \geq 0} \zeta_k^{t+(n-2k-2)/2} p_t(\zeta') \mu(\zeta) d\zeta}{\int \sum_{t\geq 0} \zeta_k^{t+(n-2k-2)/2} p_t(\zeta') \mu(\zeta) d\zeta}$$
$$= \frac{\sum_{t\geq 0} \int_{\prod_i (1-\zeta_i) \leq \epsilon^2} \zeta_k^{t+(n-2k-2)/2} p_t(\zeta') \mu(\zeta) d\zeta}{\sum_{t\geq 0} \int \zeta_k^{t+(n-2k-2)/2} p_t(\zeta') \mu(\zeta) d\zeta}$$
$$\leq \max_{t\geq 0} \frac{\int_{\prod_i (1-\zeta_i) \leq \epsilon^2} \zeta_k^{t+(n-2k-2)/2} p_t(\zeta') \mu(\zeta) d\zeta}{\int \zeta_k^{t+(n-2k-2)/2} p_t(\zeta') \mu(\zeta) d\zeta}$$
$$\leq \max_{t\geq 0} \frac{\int_{(1-\zeta_k) \leq \epsilon^{2/k}} \zeta_k^{t+(n-2k-2)/2} p_t(\zeta') \mu(\zeta) d\zeta}{\int_{(1-\zeta_k) \leq \rho} \zeta_k^{t+(n-2k-2)/2} p_t(\zeta') \mu(\zeta) d\zeta}.$$

Suppose that μ is the density for a ρ-perturbation of some fixed subspace S, given as the span of the first $(k+1)$ columns of an orthogonal matrix Q_0. For fixed values of $\zeta_1, \ldots, \zeta_{k-1}$, U, V, W, and Z, changing ζ_k by ϕ causes $\text{dist}(Q^{-1}Q_0, \text{Id})$ to change by at most ϕ. By our definition of a ρ-perturbation, this implies that

$$\mu(\zeta_1, \ldots, \zeta_{k-1}, \zeta_k) \geq \mu(\zeta_1, \ldots, \zeta_{k-1}, \zeta_k + \phi)/e$$

for any $\phi < \rho$. Consequently there exists some density $\mu'(\zeta_1, \ldots, \zeta_{k-1})$ dependent only on the first $k-1$ of the ζ_i such that

$$\mu'(\zeta')/e \leq \mu(\zeta) \leq \mu'(\zeta').$$

This allows us to bound our probability by

$$\begin{aligned}
&\max_{t\geq 0} \frac{e\int_{(1-\zeta_k)\leq\epsilon^{2/k}} \zeta_k^{t+(n-2k-2)/2} p_t(\zeta')\mu'(\zeta')d\zeta}{\int_{(1-\zeta_k)\leq\rho} \zeta_k^{t+(n-2k-2)/2} p_t(\zeta')\mu'(\zeta')d\zeta} \\
&= \max_{t\geq 0} \frac{e\int_{(1-\zeta_k)\leq\epsilon^{2/k}} \zeta_k^{t+(n-2k-2)/2} d\zeta_k}{\int_{(1-\zeta_k)\leq\rho} \zeta_k^{t+(n-2k-2)/2} d\zeta_k} \\
&= e\max_{t\geq 0} \frac{\zeta_k^{t+(n-2k)/2}\Big|_{\zeta_k=1-\epsilon^{2/k}}^{1}}{\zeta_k^{t+(n-2k)/2}\Big|_{\zeta_k=1-\rho}^{1}} \\
&\leq e\frac{1-(1-\epsilon^{2/k})^{(n-2k)/2}}{1-(1-\rho/2)^{(n-2k)/2}} \\
&\leq O(1)\frac{(n-2k)\epsilon^{2/k}}{(n-2k)\rho}, \quad \text{as } \rho < 1/(n-2k) \\
&= O(1)\frac{\epsilon^{2/k}}{\rho},
\end{aligned}$$

where the maxima in all of the above expressions are taken over the set of t for which $p_t \neq 0$.

■ 4.3.4 Putting the steps together

In Section 4.3.1, we showed that every k-dimensional facet of P has volume of at least $1/k!$. We then showed in Section 4.3.3 that

$$\mathbf{Pr}_{S\in\mathrm{Gr}(n,k+1)}\left[\frac{\mathrm{Vol}_k(\pi_S(F))}{\mathrm{Vol}_k(F)} \leq \epsilon \,\middle|\, A_F(S)\right] \leq O(1)\frac{\epsilon^{2/k}}{\rho}.$$

If Q is the projection of P onto S, combining the two results above shows that the probability that any given k-face of Q has volume less than $\epsilon/k!$, contingent upon this face appearing, is bounded above by $O(1)\epsilon^{2/k}/\rho$. This implies that the expected volume of any such face, contingent upon it appearing, is at least $(1/k!)(\rho/O(1))^{k/2}$.

The polytope Q is contained in a ball of radius $\gamma\sqrt{k}$. Its surface area is therefore bounded above by the surface area of this ball, which equals

$$2\frac{\pi^{(k+1)/2}}{\Gamma((k+1)/2)}(\gamma\sqrt{k})^k.$$

The surface area of Q equals the sum of the areas of its facets, so we obtain:

$$\begin{aligned}
&2\frac{\pi^{(k+1)/2}}{\Gamma((k+1)/2)}(\gamma\sqrt{k})^k \\
&\geq \sum_F \mathbf{E}[\mathrm{Vol}_k(\pi_S(F))|A_F(S)] \cdot \Pr[A_F(S)] \\
&\geq (1/k!)(\rho/O(1))^{k/2}\sum_F \Pr[A_F(S)] \\
&= (1/k!)(\rho/O(1))^{k/2}\,\mathbf{E}[\text{number of facets of } Q].
\end{aligned}$$

Combining this with the fact that each facet of Q contains only polynomially many vertices (since k is a constant) yields Lemma 4.3.2. $\square$

4.4 Approximation algorithms

Here we show that the polynomial smoothed bounds from Section 4.3 can be used to obtain a randomized fully-polynomial additive approximation scheme for minimizing low-rank concave functions under certain conditions. Since the objective function may have a zero minimum, the concept of multiplicative approximation is not so meaningful. On the other hand, the additive approximation implies an arbitrarily good multiplicative approximation for functions bounded below by a constant or an inverse polynomial. In the next Chapter we will show that no polynomial smoothed bounds are possible for general concave minimization.

Theorem 4.4.1. *There is a randomized fully-polynomial time algorithm with additive approximation ϵ for any $\epsilon > 0$, for low-rank quasi-concave minimization over a polynomially-bounded polytope, when the objective function satisfies a Lipschitz condition with respect to the L_1-norm*[2] *with a polynomially bounded Lipschitz coefficient $\varphi(n)$.*

Proof. Denote the given function f, then the Lipschitz condition implies

$$|f(x) - f(y)| \leq \varphi(n)|x - y|_1 \qquad \forall x, y.$$

Further, the points in the feasible set satisfy $|x|_1 \leq \psi(n)$ some polynomial ψ, where $|.|_1$ denotes the L_1-norm.

Consider the following randomized polynomial-time algorithm for minimizing f.

```
Repeat $n$ times:
  Perturb the function $f$ to a new function $\bar{f}$.
  Run an algorithm for finding the minimum of $\bar{f}$ for $2\Delta$ steps, where $\Delta$ is
    the expected smoothed polynomial bound for finding the minimum of $f$.
  If the algorithm terminates, return the minimum of $\bar{f}$.
```

With probability at least $1 - 1/2^n$, this algorithm will output $\bar{f}_{min} = \bar{f}(y) = f(Ry)$, which we will show is very close to f_{min}. By definition of the perturbation matrix R and the Lipschitz condition,

$$\begin{aligned}|\bar{f}(x) - f(x)| &= |f(Rx) - f(x)| \leq \varphi(n)|(R - I)x|_1 \\ &\leq \varphi(n) dist(R, I)|x|_1 \leq \varphi(n)\psi(n)\rho,\end{aligned}$$

for every x in the feasible set, where $dist(R, I) \leq \rho$ is the perturbation size. Thus for a perturbation $\rho \leq 1/n^l$, where l is equal to the degree of the polynomial $\varphi(n)\psi(n)$ plus $\frac{\log(1/\epsilon)}{\log n}$, we have that $|\bar{f}(x) - f(x)| \leq \epsilon$ for all x and, in particular, $\bar{f}_{min} \leq f_{min} + \epsilon$. $\square$

[2]We use the L_1-norm to facilitate the corollary below; however, the theorem holds for more general metrics.

Note that for functions bounded below by 1, the above theorem at once implies a randomized fully-polynomial approximation scheme (with multiplicative approximation factor $1 + \epsilon$).

In the next chapter, with the help of Theorem 4.4.1, we will establish that there is no smoothed polynomial-time algorithm for general quasi-concave minimization, unless $RP \neq NP$ (Corollary 5.2.4). From the analysis of Theorem 4.4.1, this implies that no polynomial smoothed bounds for *general* concave minimization are possible.

4.5 Efficient algorithms for zonotopes

In this section, we briefly mention a class of nonconvex programs which admit efficient exact algorithms, to contrast with our approximation algorithms above and the inapproximability results from Chapter 5. The programs here have zonotope feasible sets. A zonotope is the image of a hypercube under a linear transformation. Without loss of generality we will discuss maximization problems. All quasiconvex functions over a compact convex set achieve a maximum at the extreme points of the set, by Theorem 15.1.3 in the Appendix. Clearly, the class of functions with this property is wider; for example, a function with a unique critical point, which is a saddle point, also achieves a maximum at the extreme points of a compact (closed) convex set. The results in this section (as in the rest of the chapter) hold for all functions which attain an optimum at the extreme points of a compact convex set.

Consider vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$. We would like to maximize a function of $\mathbf{a}^T\mathbf{x}$ and $\mathbf{b}^T\mathbf{x}$,

$$\begin{array}{ll} \text{maximize} & g(\mathbf{a}^T\mathbf{x}, \mathbf{b}^T\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in [0,1]^n. \end{array} \tag{4.2}$$

This is equivalent to maximizing a function of two real variables, $y = \mathbf{a}^T\mathbf{x}$ and $z = \mathbf{b}^T\mathbf{x}$, which are constrained to lie on the projection of the hypercube onto the vectors $\mathbf{a}$ and $\mathbf{b}$ (we will refer to the projection as the hypercube *shadow*).

$$\begin{array}{ll} \text{maximize} & f(y,z) \\ \text{subject to} & y = \mathbf{a}^T\mathbf{x} \\ & z = \mathbf{b}^T\mathbf{x}, \quad \text{for } \mathbf{x} \in [0,1]^n. \end{array} \tag{4.3}$$

Written in this form, the constraint on y and z is a well-defined region on the plane, whose boundaries we can compute efficiently. In particular, it is the projection of the hypercube on the plane $span(\mathbf{a}, \mathbf{b})$, where each point $\mathbf{x}$ of the hypercube maps onto $(\mathbf{a}^T\mathbf{x}, \mathbf{b}^T\mathbf{x})$.

Proposition 4.5.1. *The projection of the unit hypercube in $\mathbb{R}^n$ onto $span(\mathbf{a}, \mathbf{b})$ is a convex polygon with at most $2n$ vertices, given in clockwise order by*

$$0,\ \mathbf{v}_1,\ \mathbf{v}_1 + \mathbf{v}_2, \ldots, \sum_{i=1}^{n-1} \mathbf{v}_i,\ \sum_{i=1}^{n} \mathbf{v}_i,\ \sum_{i=2}^{n} \mathbf{v}_i, \ldots, \sum_{i=n-1}^{n} \mathbf{v}_i,\ \mathbf{v}_n, \tag{4.4}$$

where $\mathbf{v}_i = (a_{i_j}, b_{i_j})$ for $j = 1, \ldots, n$ and $\frac{|a_{i_1}|}{b_{i_1}} \leq \frac{|a_{i_2}|}{b_{i_2}} \leq \ldots \leq \frac{|a_{i_n}|}{b_{i_n}}$.

Proof. Each point in the hypercube is a vector sum[3] of the n unit segments $[0, \mathbf{e}_i]$ where $\mathbf{e}_i$ is the unit vector in the i-th dimension. The projection of the hypercube equals the vector sum of the projections of these n basis segments in $span(\mathbf{a}, \mathbf{b})$, namely the segments with endpoints $(0, 0)$ and (a_i, b_i).

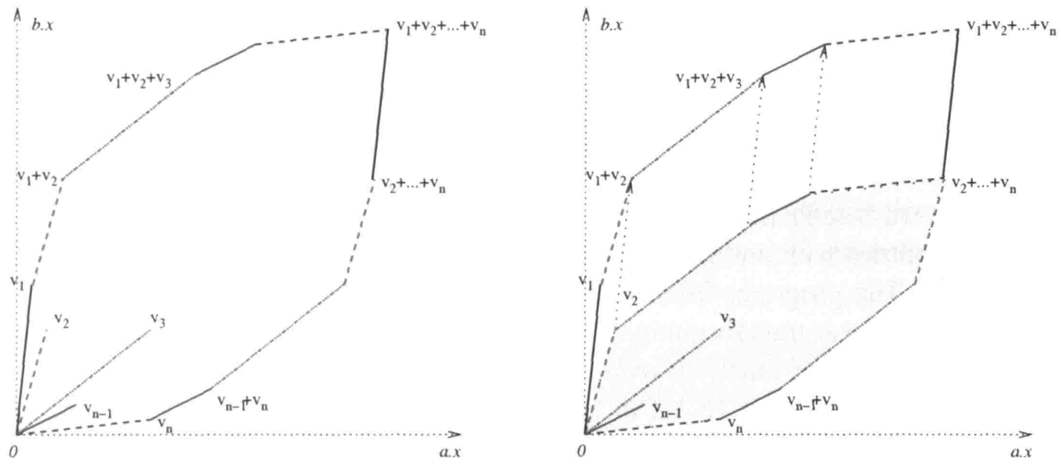


Figure 4.2. Projection of n dimensional hypercube onto $span(\mathbf{a}, \mathbf{b})$. The n unit vectors along the n dimensions project onto $\mathbf{v}_1, ..., \mathbf{v}_n$, ordered by decreasing slope $\frac{b_i}{|a_i|}$.

Next, we prove that the boundary of the projection is defined by the vertices in (4.4), by induction on n. Sort the projections of the standard basis unit vectors of $\mathbb{R}^n$ onto $span(\mathbf{a}, \mathbf{b})$ by decreasing slope, denoted in this order by $\mathbf{v}_1, ..., \mathbf{v}_n$ as in Figure 4.2. For convenience, we will use $\mathbf{v}_i$ to denote both the point and the corresponding vector. It is easy to see that the vector sum of two segments with common end at the origin and two other endpoints $\mathbf{u}$ and $\mathbf{v}$, is the resulting parallelogram whose fourth vertex is $\mathbf{u} + \mathbf{v}$. In particular, if the $[0, \mathbf{u}]$ segment is the steeper one, the vector sum lies entirely below the $[0, \mathbf{u}]$ and $[\mathbf{u}, \mathbf{u} + \mathbf{v}]$ segments, and above the $[0, \mathbf{v}]$ and $[\mathbf{v}, \mathbf{u} + \mathbf{v}]$ segments. Suppose that the statement is true for $n - 1$ segments, *i.e.*, the vector sum of $[0, \mathbf{v}_2], ..., [0, \mathbf{v}_n]$ is the polygon with vertices

$$0,\ \mathbf{v}_2,\ \sum_{i=2}^{3} \mathbf{v}_i, ...,\ \sum_{i=2}^{n-1} \mathbf{v}_i,\ \sum_{i=2}^{n} \mathbf{v}_i,\ \sum_{i=3}^{n} \mathbf{v}_i, ...,\ \sum_{i=n-1}^{n} \mathbf{v}_i,\ \mathbf{v}_n.$$

By associativity of the vector sum, the sum of all n segments is equal to the sum of $[0, \mathbf{v}_1]$ and the vector sum of the remaining $n - 1$ segments, depicted by the shaded area in Figure 4.2. The sum of $[0, \mathbf{v}_1]$ with the shaded polygon is the shaded polygon plus the new area resulting from translation of the polygon by the vector $\mathbf{v}_1$. Since $[0, \mathbf{v}_1]$ is the steepest segment, this effectively translates the upper boundary of the shaded polygon by $\mathbf{v}_1$ so the new boundary is

$$0,\ \mathbf{v}_1,\ \mathbf{v}_1 + \mathbf{v}_2,\ \mathbf{v}_1 + \sum_{i=2}^{3} \mathbf{v}_i, ...,\ \mathbf{v}_1 + \sum_{i=2}^{n-1} \mathbf{v}_i,\ \mathbf{v}_1 + \sum_{i=2}^{n} \mathbf{v}_i,\ \sum_{i=2}^{n} \mathbf{v}_i,\ \sum_{i=3}^{n} \mathbf{v}_i, ...,\ \sum_{i=n-1}^{n} \mathbf{v}_i,\ \mathbf{v}_n,$$

[3]The vector sum, also known as the Minkowski sum of two sets A and B is the set $\{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A,\ \mathbf{b} \in B\}$. Since the dot product is distributive, $(\mathbf{a} + \mathbf{b}) \cdot \mathbf{c} = \mathbf{a} \cdot \mathbf{c} + \mathbf{b} \cdot \mathbf{c}$, so the projection of a vector sum equals the vector sum of the projections.

as desired. □

The essential fact we used above was that the hypercube in $\mathbb{R}^n$ is a vector sum of n segments. Thus, an equivalent result would hold for general zonotopes, which have an alternative definition as being the vector sum of n segments.

Corollary 4.5.2. *The projection of a zonotope defined by n segments, on $span(\mathrm{a}, \mathrm{b})$ is a convex polygon with at most $2n$ vertices, described by a similar list as in Eq. (4.4).*

Next, note that the only computation necessary to find the boundary vertices, is to sort the ratios $\frac{|a_{i_1}|}{b_{i_1}} \leq \frac{|a_{i_2}|}{b_{i_2}} \leq \ldots \leq \frac{|a_{i_n}|}{b_{i_n}}$, which can be done in time $O(n \log n)$.

Now, by Theorem 15.1.3, provided that $f(y, z)$ is quasi-convex or generally attains its maximum on extreme points of the set, it suffices to check the function along the $2n$ boundary segments determined by (4.4) in order to find the desired solution.

Example 1.
An example of a nonconvex program from the class above, similar to the one arising from the probability tail objective in Section 3.4.1 is

$$\begin{array}{ll} \text{maximize} & \dfrac{\mathrm{a}^T\mathrm{x}}{\sqrt{\mathrm{b}^T\mathrm{x}}} \\ \text{subject to} & \mathrm{x} \in \{0,1\}^n, \ x \neq 0 \end{array} \tag{4.5}$$

where $a \geq 0$ and $b > 0$.

Note that the objective is strictly increasing in $\mathrm{a}^T\mathrm{x}$ and strictly decreasing $\mathrm{b}^T\mathrm{x}$, therefore its maximum lies on the south-east boundary of the feasible set shadow in $span(\mathrm{a}, \mathrm{b})$ (See Fig. 4.2). In addition, the objective function is quasi-convex so it achieves a maximum at an extreme point. Therefore the maximum of the relaxation of problem (4.5) to the convex hull of the feasible set is attained at a vertex of the shadow boundary in $span(\mathrm{a}, \mathrm{b})$, in other words, the relaxed and the integer versions will have the same maximum. Thus, it suffices to check the n vertices on the south-east boundary of the shadow, as given by Proposition 4.5.1.

Example 2.
We can also solve a traditional convex program on the hypercube faster with the above Proposition. The following program reduces to the one in (4.5)

$$\begin{array}{ll} \text{maximize} & \dfrac{(\mathrm{a}^T\mathrm{x})^2}{\mathrm{b}^T\mathrm{x}} \\ \text{subject to} & \mathrm{x} \in \{0,1\}^n, \ \mathrm{x} \neq 0 \end{array} \tag{4.6}$$

where $\mathrm{b} > 0$. This objective is the same as in a convex program with linear constraints considered by Vandenberghe and Boyd [125], which they transform and solve as a semidefinite program.

Denote $y = \mathrm{a}^T\mathrm{x}$ and $z = \mathrm{b}^T\mathrm{x}$. Since $b > 0$, we have $z > 0$ and this problem is equivalent to finding the maximum of $|y|/\sqrt{z}$. So, for $\mathrm{a} > 0$ and consequently $y > 0$, the problem is equivalent to the one in Example 1. For general a and nonnegative b (Fig. 4.3), the problem is no longer

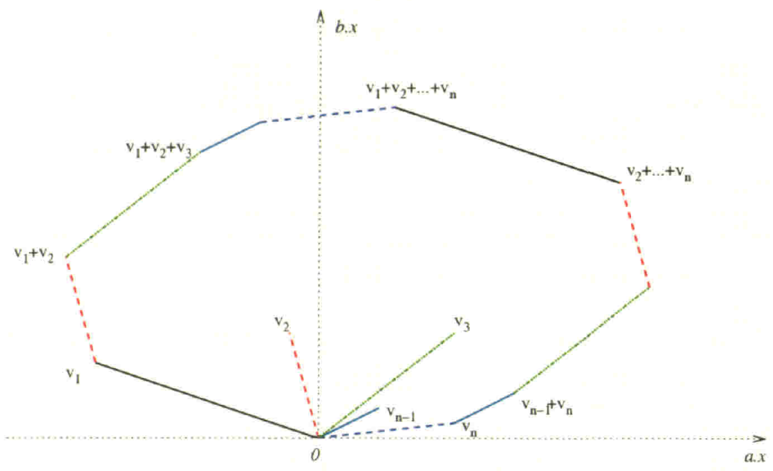


Figure 4.3. Projection of n dimensional hypercube onto $span(\mathbf{a}, \mathbf{b})$ for general $\mathbf{a}$ and nonnegative $\mathbf{b}$. The n unit vectors along the n dimensions project onto $\mathbf{v}_1, ..., \mathbf{v}_n$, ordered by decreasing slope $\frac{b_i}{|a_i|}$.

equivalent to Example 1. However, the objective is still quasi-convex since the lower level sets are convex, given by $L_\lambda = \{(y, z) \in \mathbb{R}^2 \mid z \geq y^2/\lambda^2\}$ and we can solve it by enumeration of the $2n$ extreme points on the shadow. (The objective is in fact convex; however, quasi-convexity is easier to check and suffices for our purpose.)

Finally, the efficient algorithms extend to low-rank functions of the form $g(\mathbf{a}_1^T\mathbf{x},\ \mathbf{a}_2^T\mathbf{x}, ...,\ \mathbf{a}_k^T\mathbf{x})$ where k is a constant. In this case we would project the feasible set to the k-dimensional vector space $span(\mathbf{a}_1, ..., \mathbf{a}_k)$; the number of extreme points on the projection of a hypercube, for example, would be larger but still polynomial, $O(n^k)$.

CHAPTER 5

General (high-rank) nonconvex optimization

In this chapter we describe the connection between concave and supermodular set optimization and, together with an application from stochastic optimization, we use them to establish a new result for hardness of approximation of supermodular and concave minimization. As a reference, in the last section we include prior results on hardness of approximation of submodular and convex maximization.

Submodularity has been commonly used in combinatorial optimization problems and perhaps for this reason some call it an analogue of discrete convex analysis. Indeed, computer science theorists work mainly with submodular set functions, which are defined on discrete finite sets, and Lovász [85] proved that these functions have a natural convex extension in $\mathbb{R}^n$. Since submodular functions include as a special case the cut function of a graph, maximizing submodular functions and, consequently, maximizing convex and quasi-convex functions is at least as hard as MAX CUT, a known NP-hard problem [49]. We note that the direct connections between submodularity and convexity seem to stop at submodular set functions and their corresponding convex extensions, as sketched in Figure 5.1. For example, a notion of quasi-submodularity does not extend as well to quasi-convexity. On the other hand, general submodular functions have a well-behaved set of minima similarly to general convex functions [124].

It is a fascinating open question to better understand the connections between submodularity and convexity beyond submodular set functions since general submodular functions also have important applications in Economics [92].

5.1 Correspondence of concave and supermodular optimization

We now describe a correspondence between concave (convex) and supermodular (submodular) functions due to Lovász [85], which we will use in the subsequent section to establish hardness of approximation results for concave minimization.

Submodular functions are often presented as the discrete analogues of convex functions. This is not strictly true since they can be continuous functions, *e.g.*, defined over $\mathbb{R}^m$. In general, submodular functions are defined over a lattice, that is a partially ordered set which contains the least upper bound and greatest lower bound of every pair of its elements with respect to its order relation.

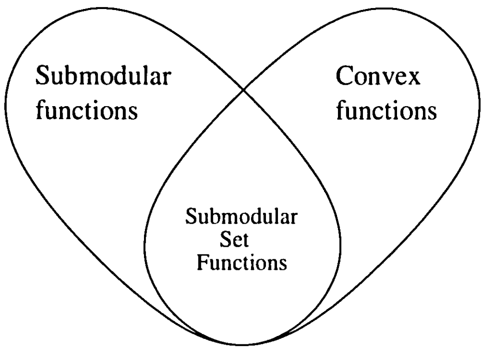


Figure 5.1. Relationship between submodular and convex functions.

Definition 5.1.1. *Let Z be a partially ordered set with order $\prec$. For $x, y \in Z$,*

(i) the join $x \vee y$ *is the least upper bound z, i.e., $x, y \prec z$ and if $x, y \prec z'$, then $z \prec z'$.*

(ii) the meet $x \wedge y$ *is the greatest lower bound of x and y.*

Z is a lattice *if it contains the join and meet of every pair of its elements.*

One example of a lattice is $\mathbb{R}^m$ with $\prec$ being the coordinate-wise relation $<$. As another example, perhaps the simplest lattice in $\mathbb{R}^m$ is the set of vertices of the unit hypercube $\{0,1\}^m \in \mathbb{R}^m$, namely the set of all 01-vectors. From a different perspective, all 01-vectors represent the subsets of some ground set $S = \{s_1, ..., s_m\}$, with 1 in the i-th coordinate if s_i is in the subset. Thus, the power set $\mathcal{P}_S$ of the set S is also a lattice, sometimes called the lattice of subsets, where the join of two subsets is their union and the meet is their intersection.

Definition 5.1.2. *A function $f : Z \longrightarrow \mathbb{R}$ over a lattice Z is submodular if*

$$f(x \vee y) + f(x \wedge y) \leq f(x) + f(y), \qquad \forall x, y \in Z. \tag{5.1}$$

In particular, f is called a submodular *set* function when it is defined over the lattice of subsets, with

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B), \qquad \forall A, B \in Z. \tag{5.2}$$

We now describe the Lovász extension of a submodular set function, which is used to define the corresponding convex/concave function.

Let χ_A denote the 01-incidence vector of the subset A. Any other nonnegative vector c can be represented uniquely as

$$c = \sum_i \lambda_i \chi_{A_i},$$

where $\lambda_i > 0$ and $A_1 \subset A_2 \subset ...$ are distinct subsets.

For a submodular set function f, define the Lovász extension $\hat{f} : \mathbb{R}^m_+ \longrightarrow \mathbb{R}$ by

$$\hat{f}(c) = \begin{cases} f(A) & \text{for } c = \chi_A \\ \sum_i \lambda_i \hat{f}(\chi_{A_i}) & \text{for } c = \sum_i \lambda_i \chi_{A_i}. \end{cases} \tag{5.3}$$

The Lovász extension seems to have a very natural definition; however, we remark that it crucially relies on the special basis $\chi_{A_1} \leq \chi_{A_2} \leq ...$ of 01-vectors and the uniqueness of representation in this basis. Note also that this basis may be different for different vectors $c \in \mathbb{R}^m_+$. For example, although the vector $(1, 1, 0, ..., 0)$ is a sum of $(1, 0, 0, ..., 0)$ and $(0, 1, 0, ..., 0)$, it cannot be represented as a positive combination of any *increasing* sequence of 01-vectors other than itself.

Lemma 5.1.3. *[85] The set function f is submodular if and only if its Lovász extension $\hat{f}$ is convex.*

Proof. Lovász's proof is more subtly presented among a number of other results in his original paper [85]; here we offer a more direct and detailed version.

($\Rightarrow$) For the forward direction, we establish a special property of submodular set functions, which allows us to represent their Lovász extension as the supremum of linear functions, thereby concluding that the extension is convex.

As before, denote the ground set of all elements S. We are going to show that for every nonnegative c,

$$\hat{f}(c) = \max\{c \cdot x \mid x(A) \leq f(A) \quad \forall A \subset S\},$$

where $x \in \mathbb{R}^m$ and $x(A) = x \cdot \chi_A$, that is $x(A)$ is the sum of those coordinates of x corresponding to elements in the subset A. Consider the linear program above and its dual,

PRIMAL	DUAL
$\max\ c \cdot x$	$\min \sum_{A \subset S} f(A) y_A$
s.t. $x(A) \leq f(A) \quad \forall A \subset S$	s.t. $\sum_{i \in A \subset S} y_A = c_i$
	$y_A \geq 0$

Let $c = \sum_i \lambda_i \chi_{A_i}$, where $\lambda_i \geq 0$ and $A_1 \subset A_2 \subset ... \subset A_m$ being distinct subsets. (If the unique basis of c contains less than m basis subsets A_i with positive λ_i, we can complete them to be exactly m with the remaining subsets having $\lambda_j = 0$.) Since A_i are all distinct and nested, it must be that A_i has exactly i elements; without loss of generality, say

$$\begin{aligned} A_1 &= \{s_1\} \\ A_2 &= \{s_1, s_2\} \\ &\vdots \\ A_m &= \{s_1, s_2, ..., s_m\}. \end{aligned}$$

With this ordering, $c(s_1) = \sum_{i=1}^m \lambda_i$, $c(s_2) = \sum_{i=2}^m \lambda_i$, etc. where $c(s_i)$ is the coordinate of c whose index is the original index of the element s_i. Thus, $c(s_1) \geq c(s_2) \geq ... \geq c(s_m)$.

Since the Primal and Dual have exponentially many inequalities and variables respectively, they cannot be solved directly; however, we can derive an optimal solution by "guessing" feasible primal and dual solutions with matching objective values. (Or, the two solutions can be thought of as the result of a greedy algorithm, which magically works because of the submodularity property of f.) Consider the primal solution x^* with $x^*(s_k) = f(A_k) - f(A_{k-1})$. We claim that it is feasible; for this we need to show that $x^*(T) = \sum_{i:\chi_T(i)=1} f(A_i) - f(A_{i-1}) \leq f(T)$. Suppose $T = \{s_{i_1}, ..., s_{i_p}\}$. By submodularity of f,

$$\begin{aligned} f(T) &\geq f(A_{i_1-1} \cup T) - f(A_{i_1-1}) \\ &\geq [f(A_{i_2-1} \cup T) + f(A_{i_1}) - f(A_{i_2-1})] - f(A_{i_1-1}) \\ &\vdots \\ &\geq [f(A_{i_p-1} \cup T) + f(A_{i_{p-1}}) - f(A_{i_p-1})] - f(A_{i_{p-1}-1}) + \sum_{k=1}^{p-2} [f(A_{i_k}) - f(A_{i_k-1})] \\ &= \sum_{k=1}^{p} [f(A_{i_k}) - f(A_{i_k-1})] = x(T), \end{aligned}$$

where we repeatedly use the submodularity relation $f(A) \geq f(A \cup B) + f(A \cap B) - f(B)$ with $A = T$, $B = A_{i_1-1}$ for the first inequality and $A = A_{i_{k-1}-1} \cup T$, $B = A_{i_k-1}$ for the k-th inequality, for $k = 2, ..., p$. This concludes the proof of feasibility of x^*. The corresponding objective function value is $c \cdot x^* = \sum_{k=1}^{m} c(s_k)[f(A_k) - f(A_{k-1})]$.

Next, consider the dual solution

$$y_T^* = \begin{cases} c(s_k) - c(s_{k+1}) & \text{if } T = A_k, \text{ for } k = 1, 2, ..., m-1 \\ c(s_m) & \text{if } T = A_m \\ 0 & \text{otherwise.} \end{cases}$$

Since $c(s_1) \geq ... \geq c(s_m) \geq 0$, we have $y_T \geq 0$. In addition,

$$\sum_{T \ni s_i} y_T^* = \sum_{k=i}^{m} [c(s_k) - c(s_{k+1})] = c(s_i).$$

This proves feasibility of y_T^*, and the corresponding dual objective value is

$$\sum_{T} f(T) y_T^* = \sum_{k=1}^{m} f(A_k) y_{A_k}^* = \sum_{k=1}^{m} c(s_k)[f(A_k) - f(A_{k-1})],$$

where $f(A_0) = f(\oslash)$. Thus, the primal and dual objectives are equal, hence x^* and y^* are optimal primal and dual solutions. Finally,

$$c = \sum_{i=1}^{m} c(s_i)[\chi_{A_i} - \chi_{A_{i-1}}] = \sum_{i=1}^{m} [c(s_i) - c(s_i - 1)]\chi_{A_i} = \sum_{i=1}^{m} y_{A_i}^* \chi_{A_i}$$

therefore $y_{A_i}^* = \lambda_i$ and by definition of the Lovász extension,

$$\hat{f}(c) = \sum_{i=1}^{m} y_{A_i}^* f(A_i) = \max\{c \cdot x \mid x(T) \leq f(T), \forall T \subset S\}.$$

Therefore, $\hat{f}$ is the supremum of linear functions over the polyhedron $\{x(T) \leq f(T), \forall T \subset S\}$, so it is convex.

($\Leftarrow$) Conversely, suppose the Lovász extension $\hat{f}$ is convex. Then,

$$\begin{aligned} f(A \cup B) + f(A \cap B) &= \hat{f}(\chi_{A\cup B}) + \hat{f}(\chi_{A\cap B}) \\ &= \hat{f}(\chi_{A\cup B} + \chi_{A\cap B}) \\ &= \hat{f}(\chi_A + \chi_B) \\ &\leq \hat{f}(\chi_A) + \hat{f}(\chi_B) = f(A) + f(B). \end{aligned}$$

The second equality follows from the definition of Lovász extension. The inequality follows from $\hat{f}(\gamma c) = \gamma \hat{f}(c)$, by the definition of Lovász extension and consequently $\frac{1}{2}\hat{f}(\chi_A + \chi_B) = \hat{f}(\frac{1}{2}\chi_A + \frac{1}{2}\chi_B) \leq \frac{1}{2}\hat{f}(\chi_A) + \frac{1}{2}\hat{f}(\chi_B)$ by convexity of $\hat{f}$. $\square$

5.2 Hardness of approximation of concave minimization

In this section, we prove that minimizing a supermodular function and consequently, minimizing a (quasi)concave function, is $(\log n)$-hard to approximate. The proof is by a gap-preserving reduction from the two-stage stochastic Minimum Spanning Tree (MST) problem considered by Flaxman, Frieze and Krivelevich [42], which is $(\log n)$-hard to approximate [42].

The two-stage stochastic MST problem is defined as follows: Edges in the graph have given weights in the first stage and random weights from known distributions in the second stage. The goal is to choose an optimal set of edges in the first stage that can be completed with edges in the second stage to form a spanning tree of minimum expected cost. More formally, denote the cost of edge e in the first and second stages by $C_1(e)$ and $\mathcal{C}(e)$ respectively. Note that a subset of edges S chosen in the first stage uniquely determines (up to equal cost) the set of edges T_S chosen in the second stage which complete the MST. Our problem is to compute $\min_{S\subset E}\left\{h(S) = C_1(S) + \mathbf{E}[\mathcal{C}(T_S)]\right\}$, where E is the set of edges of the graph. We will show that the function $h(S)$ is supermodular. To simplify notation, we shall use the shorthand $S + e := S \cup \{e\}$ and $S - e := S \backslash \{e\}$. We also use $A \subset B$ to mean non-strict set inclusion.

Lemma 5.2.1. *The function $h(S) = C_1(S) + \mathbf{E}[\mathcal{C}(T_S)]$ is supermodular.*

Proof. The function h is supermodular if and only if the marginal contribution of an edge to the function value is bigger whenever the edge is added to a bigger set [85], that is

$$h(A+e) - h(A) \leq h(B+e) - h(B)$$

for all $A \subset B \subset E$ and $e \in E$, $e \notin A, B$. Assume without loss of generality that A, B, as well as $A+e, B+e$ do not contain cycles. We have $C_1(A+e) - C_1(A) = C_1(e) = C_1(B+e) - C_1(B)$. By linearity of expectation, it suffices to show that

$$\mathcal{C}(T_{A+e}) - \mathcal{C}(T_A) \leq \mathcal{C}(T_{B+e}) - \mathcal{C}(T_B),$$

or equivalently

$$\mathcal{C}(T_B) - \mathcal{C}(T_{B+e}) \leq \mathcal{C}(T_A) - \mathcal{C}(T_{A+e}), \tag{5.4}$$

for every realization of the second stage costs $\mathcal{C}$. Without loss of generality up to equality of the tree costs we can assume that $T_B \subset T_A$ since $A \subset B$. We now need to consider several cases for the edge e.

(i) $e \in T_A, T_B$. Then both sides of the inequality (5.4) are zero.

(ii) $e \in T_A, e \notin T_B$. Since $(A+e) \cup (T_A - e) = A \cup T_A$ is a valid spanning tree and $(A+e) \cup T_{A+e}$ is the minimal spanning tree corresponding to choice set $A + e$ from the first stage, we have $\mathcal{C}(T_{A+e}) \leq \mathcal{C}(T_A - e) = \mathcal{C}(T_A) - \mathcal{C}(e)$. Similarly, $\mathcal{C}(T_B) \leq \mathcal{C}(T_{B+e} + e)$ since $B \cup T_B$ is the cheapest spanning tree which contains set B and $B \cup (T_{B+e} + e) = (B + e) \cup T_{B+e}$ is another spanning tree containing set B. Therefore

$$\mathcal{C}(T_B) - \mathcal{C}(T_{B+e}) \leq \mathcal{C}(e) \leq \mathcal{C}(T_A) - \mathcal{C}(T_{A+e}),$$

and we are done.

(iii) $e \notin T_A, T_B$. Without loss of generality $T_{A+e} \subset T_A$ and $T_{B+e} \subset T_B$. Let $T_{A+e} = T_A - e'$ and $T_{B+e} = T_B - e''$ where e' and e'' are the heaviest edges in the cycles formed in the spanning trees $A \cup T_A$ and $B \cup T_B$ respectively when edge e is added. (Note: this assumes that $A + e$, $B + e$ do not contain cycles so any cycles formed would be in T_A, T_B.) Since $T_B \subset T_A$, the heaviest edge e'' in the cycle in T_B is no heavier than e' in the cycle in T_A. Therefore,

$$\begin{aligned}\mathcal{C}(T_B) - \mathcal{C}(T_{B+e}) &= \mathcal{C}(e'') \leq \mathcal{C}(e') \\ &= \mathcal{C}(T_A) - \mathcal{C}(T_{A+e}).\end{aligned}$$

This completes the proof. □

We emphasize that the proof of Lemma 5.2.1 does not depend in any way on the distributions of the second stage costs; in particular, they can be correlated or independent. (The dependency requirement is crucial in the proof that the two-stage MST problem is $(\log n)$-hard to approximate [42].) The lemma thus gives the desired gap-preserving reduction for supermodular minimization from the two-stage MST problem; this may be of independent interest to the study of two-stage stochastic optimization as it provides a connection between the latter and (non-monotone) supermodular and concave minimization.

Lemma 5.2.2. *Supermodular minimization is* $(\log n)$*-hard to approximate, assuming* $P \neq NP$.

Next, we show that quasi-concave minimization is also hard to approximate.

Theorem 5.2.3. *Quasi-concave minimization is* $(\log n)$*-hard to approximate, assuming* $P \neq NP$.

Proof. Given a supermodular set function h defined on the vertices of the unit hypercube in $\mathbb{R}^n$ (which we also call 0-1-vectors), its Lovász extension $f : \mathbb{R}^n_+ \to \mathbb{R}$ is defined as $f(x) = h(\mathbf{0}) + \sum_{i=1}^n \lambda_i h(b_i)$, where $\mathbf{0}$ is the zero vector and $x = \sum_{i=1}^n \lambda_i b_i$ is the unique representation of x via a basis of increasing 0-1-vectors $b_1 < b_2 < ... < b_n$ and $\lambda_i \geq 0$. Lovász showed that h is supermodular if and only if its extension f is concave [85].

We will show that there is a gap-preserving reduction of (quasi-)concave minimization from supermodular minimization.

By Theorem 15.1.3, the minimum of a supermodular set function and the minimum of its Lovász extension over the unit hypercube will coincide. Therefore, a γ-approximation of the minimum of a supermodular function is also a γ-approximation of the minimum of its corresponding Lovász extension.

Conversely, suppose we can approximate the minimum f_{min} of the Lovász extension f over the unit hypercube within a factor of γ, namely we can find $x \in [0,1]^n$ such that $f(x) \leq \gamma f_{min}$. It follows by the Caratheodory/Krein-Milman theorem [11] that x is a convex combination of vertices of the hypercube, namely $x = \sum_i \lambda_i x_i$ where x_i are 0-1-vectors and $\sum_i \lambda_i = 1$, $\lambda_i > 0$. Since f is concave, $\gamma f_{min} \geq f(x) \geq \sum_i \lambda_i f(x_i)$, therefore for at least one hypercube vertex x_j, we have $f(x_j) \leq \gamma f_{min}$. Therefore x_j would give a γ-approximation of the supermodular function minimum, as desired. □

Corollary 5.2.4. *There is no smoothed polynomial-time algorithm for general quasi-concave minimization, assuming $RP \neq NP$.*

Proof. Suppose contrarily that there is a smoothed expected polynomial-time algorithm. We will show that this implies the existence of a randomized fully polynomial approximation scheme for finding the minimum of f, the Lovász extension of the objective function h from Lemma 5.2.1, thereby contradicting theorem 5.2.3.

More precisely, we will work with the specific objective function h which is used in showing that the random 2-stage MST problem is $(\log n)$-hard to approximate [42]. We note that replacing the infinite edge costs by costs of n^3 does not alter the proof of hardness of the 2-stage MST, and thus the important property of the expected spanning tree cost h for our purpose is that it takes values in a polynomially bounded interval: $1 \leq h(x) \leq n^4$ for all 01-vectors x.

Next, we show that the Lovász extension f of h satisfies the Lipschitz condition

$$f(x) - f(y) \leq 2n^4 |x-y|_1,$$

where $|.|_1$ is the L_1 norm. By the definition of Lovász extension, f is piecewise linear, therefore the ratio

$$\frac{f(x) - f(y)}{|x-y|_1}$$

is maximized when x, y are both on one of the linear segments $f_0(x) = \sum_i c_i x_i$. Again from the definition of f, we have $c_1 = f(b_{j_1})$ and $c_i = f(b_{j_i}) - f(b_{j_{i-1}})$ for $i = 2, ..., n$ where $\{b_{j_1}, ..., b_{j_n}\}$ is the basis for representing x in the Lovász extension definition. Therefore $|c_i| \leq 2n^4$.

Finally,

$$\frac{f(x) - f(y)}{|x-y|_1} = \frac{\sum_i c_i x_i - \sum_i c_i y_i}{\sum_i |x_i - y_i|} \leq \max_i |c_i| \leq 2n^4.$$

Now applying Theorem 4.4.1 implies the result and we get the desired contradiction. □

The last result complements the existence of a randomized fully-polynomial time approximation algorithm for low-rank quasi-concave minimization established in Theorem 4.4.1. In particular, it implies that no polynomial smoothed bounds for general concave minimization are possible.

5.3 Hardness of approximation of convex maximization

To complement and put in context our results in the previous section on the hardness of concave minimization, we include here known results about the hardness of submodular maximization, which analogously to the proof of Theorem 5.2.3, yield equivalent hardness for (quasi)convex maximization.

Intuitively, in function optimization it should not matter whether we are minimizing function f or maximizing function $-f$: we expect to get equivalent results. In combinatorial optimization, on the other hand, the change of optimization direction is critical: `Min Cut` can be solved efficiently while `Max Cut` is NP-hard; similarly `Shortest path` is easy while `Longest path` is NP-hard, as is the case of many other classical combinatorial problems. This does not contradict the intuition above. In combinatorial optimization instances the minimum and maximum are different (opposite) vertices of the feasible polytope: it is typical that one side of the polytope is easily computable and the other is not. For the same reason most such problems critically depend on the linear objective function having positive coordinates (*i.e.,* the ground set of elements having positive weights).

In combinatorial optimization, the standard solution concept is that of *multiplicative* approximation. This is perhaps justified when the objective functions are linear and the main challenge in the problem lies in the combinatorial feasible set. It may be less meaningful in function approximation when the objectives are nonlinear functions. For example, a vertical shift of the function would maintain the same optimum and does not change the structure of the problem. However, the multiplicative approximation guarantee changes with the shift and can get arbitrarily bad or even meaningless (when the function minimum is negative but the next best solution has a positive function value).

In this section and the entire dissertation we follow the standard of multiplicative approximation. (Except in some of our low-rank quasiconcave minimization algorithms in Section 4.4 which more naturally provide an additive approximation.) We leave as an open philosophical and research problem what approximative concepts may be more meaningful—and the answer may well differ based on the specific application and user needs.

We first describe the straightforward fact that MAX CUT is a special case of submodular maximization since the cut of a graph is a submodular set function.

Definition 5.3.1. *Consider an undirected graph G with vertex set V. A* cut *of the graph is a partition of the vertices into two sets $\{S, V - S\}$, for $S \subset V$. The* cut function *is defined on all cuts of the graph and is equal to the number of edges crossing the cut $\{S, V - S\}$.*

Since the first set S uniquely defines the cut $\{S, V - S\}$, the cut function is a function over the subsets of V.

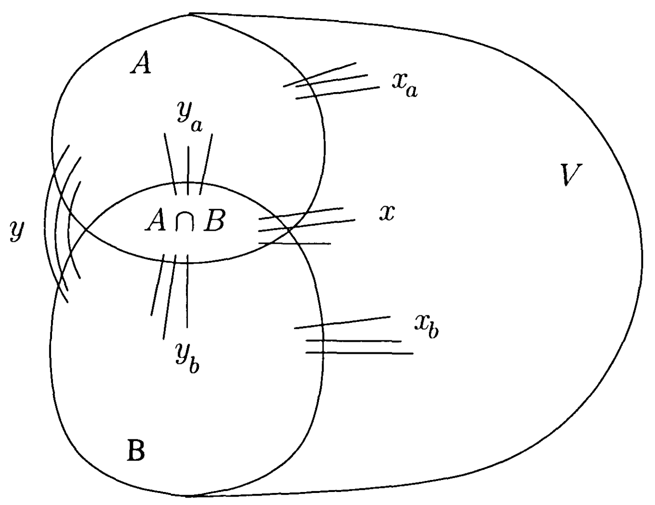


Figure 5.2. Cuts in a graph.

Lemma 5.3.2. *The cut function $f : \mathcal{P}_V \to \mathbb{R}$ is a submodular set function.*

Proof. Consider two cuts $\{A, V - A\}$ and $\{B, V - B\}$. They divide the vertex set V into four disjoint subsets $A - B$, $B - A$, $A \cap B$ and $V - A - B$, depicted in Figure 5.2. (For general sets A, B we use $A - B = A \setminus (A \cap B)$ to mean the elements from A which are not in B). Denote the number of edges crossing the different pairs of subsets as shown in the figure, $\{A - B, V - A - B\}$ by x_a, $\{B - A, V - A - B\}$ by x_b, $\{A \cap B, V - A - B\}$ by x, $\{A - B, A \cap B\}$ by y_a, $\{B - A, A \cap B\}$ by y_b and $\{A - B, B - A\}$ by y.

The cut edges from A to $V - A$ consist of all edges from $A - B$ to $B - A$ and $V - A - B$, as well as from $A \cap B$ to $V - A - B$ and $B - A$, therefore $f(A) = x_a + x + y_b + y$. Similarly, $f(B) = x_b + x + y_a + y$, $f(A \cap B) = x + y_a + y_b$ and $f(A \cup B) = x + x_a + x_b$. Therefore,

$$f(A \cup B) + f(A \cap B) = x_a + x_b + y_a + y_b + 2x \leq x_a + x_b + y_a + y_b + 2x + 2y = f(A) + f(B),$$

so $f(.)$ is submodular. □

Theorem 5.3.3. *Maximizing submodular functions is NP-hard and it is at least as hard to approximate as MAX CUT.*

Proof. By Lemma 5.3.2, MAX CUT is a special case of maximizing a submodular function. Since MAX CUT is NP-hard [49], the general problem of maximizing submodular functions is NP-hard. □

We now summarize some recent results on submodular maximization. These results assume that for a given set S, the submodular function value $f(S)$ is known (in other words, provided by an oracle—the so called *value oracle model*).

Theorem 5.3.4. *[38] Submodular set maximization in the value oracle model is $(\frac{1}{2} + \epsilon)$-hard to approximate, in the sense that for any $\epsilon > 0$, there are instances of nonnegative symmetric* *submodular maximization, such that there is no algorithm using less than $\epsilon^{\epsilon^2 n/16}$ queries that always finds a solution of expected value at least $(\frac{1}{2} + \epsilon)$ times the optimum.*

In addition, there are polynomial-time algorithms that, given a nonnegative submodular maximization instance, find a $\frac{2}{5}$-approximation.

The positive results can be improved to $\frac{1}{2}$-approximation in the case of nonnegative symmetric submodular functions [38].

The analogues between submodularity and convexity seem to end at submodular set functions, which are of course a narrow class of all submodular functions. Nevertheless, this connection is enough for establishing the hardness of maximizing convex functions. The connection is established through the Lovász extension of a submodular set function, similarly to Theorem 5.2.3.

Theorem 5.3.5. *Convex function maximization is NP-hard and it is at least as hard to approximate as submodular set function maximization.*

CHAPTER 6

Applications

6.1 Parametric optimization

In a typical combinatorial optimization problem, the goal is to find the feasible solution x with smallest weight $\mathrm{w}^T\mathrm{x}$. In parametric optimization, one needs to trade off two (or more) weights (*e.g.*, cost and weight, or length and time, etc.). In particular, we are interested in finding all solutions that minimize a positive linear combination of the two weights, $\mathrm{a}+\lambda\mathrm{b}$, where the parameter λ ranges over $[0,\infty)$. When the feasible set is combinatorial or discrete, the same feasible solution x' may minimize the objective $(\mathrm{a}+\lambda\mathrm{b})^T\mathrm{x}$ for multiple values of the parameter λ that form an interval $[\lambda_i, \lambda_{i+1}]$. We call the parameter values λ_i at which the optimal solution changes *breakpoints*.

Definition 6.1.1. *The* parametric complexity *of a parametric optimization problem is the maximum possible number of breakpoints.*

An overview and development of the parametric complexity of combinatorial problems can be found in Carstensen's thesis [19]. We summarize below some of the known results.

Theorem 6.1.2. *[Summary of Carstensen [19] and references therein] The parametric complexity of:*

1. *minimum spanning trees and more generally matroids is polynomial.*
2. *shortest path is $n^{1+\log n}$, where n is the number of nodes in the network.*
3. *parametric capacity maximum flow is at least $(2^{n/2}-2)$.*
4. *minimum cost flow is at least $2^{n/2-1}$.*
5. *linear programming $\{\min(\mathrm{c}+\lambda\mathrm{c}')^T\mathrm{x} \mid A\mathrm{x}=\mathrm{b},\ \mathrm{x}\geq 0\}$ is exponential regardless of the size of the input.*
6. *0-1 programming $\{\min(\mathrm{c}+\lambda\mathrm{c}')^T\mathrm{x} \mid A\mathrm{x}=\mathrm{b},\ x_i\in\{0,1\}\ \forall i\}$ is polynomial in n and the maximum coordinates of c, c'.*

Graphically, the optimum weight curve of a parametric problem as a function of the parameter λ is a piecewise linear function, given by the infimum of linear functions corresponding to the weights of all feasible solutions, as shown in Figure 6.1(right).

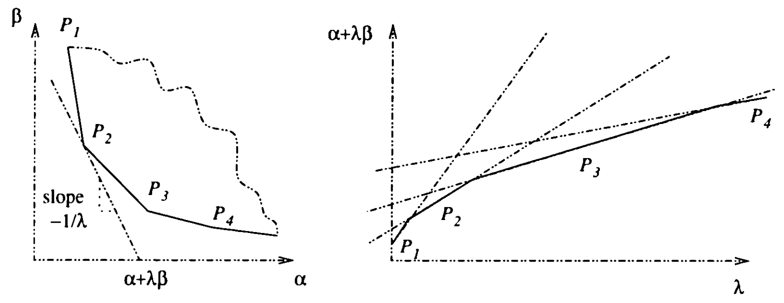


Figure 6.1. Correspondence of extreme points $P_1, P_2, \ldots$ of the dominant of the projected feasible polytope (left) and linear segments from the parametric optimal weight curve $g(\lambda) = \inf_{\mathbf{x}}\{\mathbf{a}^T\mathbf{x} + \lambda(\mathbf{b}^T\mathbf{x})\}$ (right).

An alternative graphical representation is the trade-off curve in Figure 6.1(left), with the a- and b-weights on the two axes and feasible solutions $\mathbf{x}$ represented by the points $(\mathbf{a}^T\mathbf{x}, \mathbf{b}^T\mathbf{x})$, plotting the two weights of each solution.

Clearly, this second graphical representation is the same as the one in our stochastic optimization framework from Chapter 3 with the mean and variance vectors corresponding to the $\mathbf{a}$ and $\mathbf{b}$ weights, and also represents the rank-2 nonconvex optimization from Section 4.1 with objective functions $f(\mathbf{x}) = \tilde{f}(\mathbf{a}^T\mathbf{x}, \mathbf{b}^T\mathbf{x})$. In Figure 6.1, the trade-off curve is the lower-left boundary of the projection of the feasible polytope of the stochastic and nonconvex problems onto the plane $span(\mathbf{a}, \mathbf{b})$.

The lemma below formalizes the equivalence of the parametric and stochastic / rank-2 nonconvex problems in terms of the corresponding number of their breakpoints and extreme points.

Lemma 6.1.3. *There is a one-to-one correspondence between the extreme points on the dominant of the feasible polytope projection on the plane spanned by vectors* $\mathbf{a}$, $\mathbf{b}$ *and the breakpoints of the parametric problem with weights* $\mathbf{a} + \lambda\mathbf{b}$ *over the same feasible set.*

Proof. Every extreme point on the dominant of the polytope projection onto the plane spanned by $\mathbf{a}$, $\mathbf{b}$ is the solution to a linear program:

$$\begin{array}{ll} \text{minimize} & (\mathbf{a}^T\mathbf{x}) + \lambda(\mathbf{b}^T\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \text{ feasible polytope} \end{array} \tag{6.1}$$

for a sub-interval of $\lambda \in (0, \infty)$. Plotting the objective value $g(\lambda) = \inf_{\mathbf{x}}\{\mathbf{a}^T\mathbf{x} + \lambda\mathbf{b}^T\mathbf{x}\}$ as a function of $\lambda \geq 0$ gives a concave piecewise linear function, drawn in Fig. 6.1(right). Each segment in this piecewise linear function corresponds to a feasible solution $\mathbf{x}$ with weight $\mathbf{a}^T\mathbf{x} + \lambda\mathbf{b}^T\mathbf{x}$ for an interval of the parameter λ. The feasible solutions represented by the segments are precisely the set of optimal solutions in the parametric problem. On the other hand, a solution $\mathbf{x}'$ with weight $A + \lambda B$ is optimal for a non-empty open interval of λ if and only if there is an open interval of linear objective vectors $(1, \lambda)$ which are minimized at the point (A, B) in the $span(\mathbf{a}, \mathbf{b})$-plane, namely the point (A, B) is an extreme point of the feasible polytope projection. In addition, since $\lambda > 0$, the set of optimal solutions are in one-to-one correspondence with the extreme points on the dominant of the projected feasible set, QED. □

By Lemma 6.1.3, the results for the complexity of parametric optimization problems immediately imply equivalent bounds for the number of extreme points on the dominant of the feasible set projection on a plane.

Conversely, our results bounding the number of extreme points in an average and smoothed sense imply equivalent bounds for the complexity of parametric optimization problems.

Our bounds apply to arbitrary feasible sets and so extend to general parametric optimization problems of the form $\{\min(\mathbf{a}^T\mathbf{x} + \lambda\mathbf{b}^T\mathbf{x}) \mid \mathbf{x} \in \text{ feasible set}\}$, and, in particular, to all such problems listed in Theorem 6.1.2. We state here this immediate corollary from Lemma 6.1.3 and Theorem 4.1.3.

Corollary 6.1.4. *Consider a polyhedral feasible set $\mathcal{F} \subset \mathbb{R}^n$. Suppose $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ are uniformly random unit vectors and $\lambda \in [0, \infty)$. Then the expectation of the number of optimal solutions of the parametric problem $\{\min(\mathbf{a}^T\mathbf{x} + \lambda\mathbf{b}^T\mathbf{x}) \mid \mathbf{x} \in \mathcal{F}\}$ is at most $2\sqrt{2}\pi n$.*

Similarly, a smoothed bound for parametric optimization follows from Lemma 6.1.3 and Theorem 4.1.6.

Corollary 6.1.5. *Consider a polyhedral feasible set $\mathcal{F} \subset \mathbb{R}^n$. Suppose $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ are given vectors and let $\tilde{\mathbf{a}}, \tilde{\mathbf{b}}$ be their respective ρ-perturbations as specified in Definition 4.1.5. Let also $\lambda \in [0, \infty)$. Then the expectation of the number of optimal solutions of the parametric problem $\{\min(\tilde{\mathbf{a}}^T\mathbf{x} + \lambda\tilde{\mathbf{b}}^T\mathbf{x}) \mid \mathbf{x} \in \mathcal{F}\}$ is at most $4\pi\sqrt{2n}/\rho$, for $\rho < 1/\sqrt{n}$.*

6.1.1 Parametric shortest path

The parametric complexity of combinatorial problems is most often exponential and sometimes polynomial, as can be seen from Theorem 6.1.2. The shortest path problem thus stands out with an unusual complexity that falls between polynomial and exponential. In addition, since the shortest path problem is ubiquitous and, as shown above, its parametric complexity determines the complexity of its stochastic and nonconvex variants, it is of interest to identify nontrivial cases (*e.g.*, special classes of graphs) for which the parametric complexity is polynomial. We leave this last question largely open, except for identifying such classes of graphs through an analogous application of the proof for the general parametric bound, and highlight our conjecture that the parametric complexity of shortest paths in planar graphs is polynomial.

Conjecture 6.1.6. *The parametric complexity of shortest paths in planar graphs is polynomial.*

In this section we include a simple proof of the $n^{(1+\log n)}$ upper bound for the parametric complexity of shortest paths in general graphs, and then in an effort to better understand the complexity in planar graphs, provide linear lower bounds for the latter.

Upper bound for the number of breakpoints in general graphs

In this section we show that the number of breakpoints in a general n-node graph is at most $n^{\log n+1}$.

Lemma 6.1.7. *[28] The number of breakpoints in the parametric shortest path problem is at most $n^{\log n+1}$.*

Proof. Denote by $N(k)$ the maximum number of breakpoints between any two vertices in the graph when we consider only paths of at most k edges. Then we can bound $N(2k)$ as follows: An optimal path from S to T with at most $2k$ edges consists of two subpaths from S to node v and from v to T each of at most k edges. There are n choices for v, and for each choice the optimal cost curve from S to T through v has at most $2N(k)$ breakpoints (as a sum of the optimal cost curves from S to v and from v to T).

Therefore $N(2k) \leq n[2N(k)]$ and note $N(1) \leq 1$. Hence,

$$N(2^k) \leq 2nN(2^{k-1}) \leq (2n)^2 N(2^{k-2}) \leq \ldots \leq (2n)^k N(1) \leq (2n)^k.$$

Substituting $k = \log n$ gives $N(n) \leq n^{\log n+1}$. □

The upper bound proof above was communicated to us by Dean [28]; a different more involved proof appears in Carstensen [19].

The following is an easy but fundamental property of shortest path breakpoints. It says that once an optimal subpath between nodes X, Y is replaced by a different subpath, the former expires, *i.e.,* it cannot become optimal at a later stage.

Lemma 6.1.8 (Expiration property). *Suppose the lowest-cost curve of the graph contains three breakpoints $\lambda_1 < \lambda_2 < \lambda_3$. Then it is impossible for the paths corresponding to these breakpoints to use subpaths π, π', π respectively between two internal nodes X, Y.*

Proof. Suppose the contrary. Denote the weights corresponding to π, π' by $(a + b\lambda)$ and $(a' + b'\lambda)$ respectively where $(a, b) \neq (a', b')$. Further denote the remaining weights on the three paths by $(a_1 + b_1\lambda)$, $(a_2 + b_2\lambda)$ and $(a_3 + b_3\lambda)$ respectively. Denote the subset of remaining edges for the three paths by π_1, π_2, π_3.

At λ_1, $\{\pi_1, \pi\}$ is the optimal path, therefore

$$\begin{aligned} & (a_1 + b_1\lambda_1) + (a + b\lambda_1) \leq (a_1 + b_1\lambda_1) + (a' + b'\lambda_1) \\ \Rightarrow \quad & a + b\lambda_1 \leq a' + b'\lambda_1. \end{aligned}$$

Similarly at λ_2, $\{\pi_2, \pi'\}$ is the optimal path, therefore

$$\begin{aligned} & (a_2 + b_2\lambda_2) + (a + b\lambda_2) \geq (a_2 + b_2\lambda_2) + (a' + b'\lambda_2) \\ \Rightarrow \quad & a + b\lambda_2 \geq a' + b'\lambda_2. \end{aligned}$$

Finally at λ_3, $\{\pi_3, \pi\}$ is the optimal path, therefore

$$\begin{aligned} & (a_3 + b_3\lambda_3) + (a + b\lambda_3) \leq (a_3 + b_3\lambda_3) + (a' + b'\lambda_3) \\ \Rightarrow \quad & a + b\lambda_3 \leq a' + b'\lambda_3. \end{aligned}$$

This implies that the lines $a + b\lambda$ and $a' + b'\lambda$ cross in two places or are identical, which is a contradiction. □

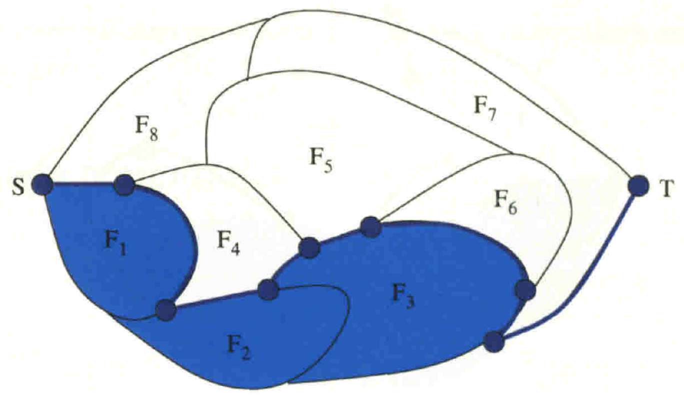


Figure 6.2. A valid face ordering and a path determined by faces $\{F_1, F_2, F_3\}$.

A lower bound for the number of breakpoints in planar graphs

In this section, we prove that for every undirected planar graph there exist edge weights such that the number of breakpoints is equal to the number of faces of the graph plus one.

Denote the weights of each edge e by a_e, b_e. For ST-paths π define the *breakpoint* or *lower envelope* function

$$B(\lambda) = \min_{\pi} \Big\{ \sum_{e\in\pi} a_e + \lambda \sum_{e\in\pi} b_e \Big\}.$$

$B(\lambda)$ is a piece-wise linear function defined on $\lambda \in [0, \infty)$; we call its points of non-differentiability breakpoints. Each linear segment $\sum_{e\in\pi} a_e + \lambda \sum_{e\in\pi} b_e$ of the function corresponds to a path which is shortest with respect to edge weights $a_e + \lambda b_e$ for some interval of λ's.

We will construct edge weights a_e, b_e such that the breakpoints are precisely at $\lambda_i = i$ for $i = 0, 1, ..., k$ where k is the number of inner faces in the planar graph.

The boundary of the (outer face of the) planar graph is divided into two by S and T; refer to the two parts as top and bottom boundary or top-most and bottom-most ST-path. Our construction uses a special ordering of the faces of the graph, which we call a valid face ordering.

Definition 6.1.9. *A* valid face ordering *of the faces of a planar graph is an ordering $F_1, F_2, ..., F_k$ such that each two consecutive faces F_i, F_{i+1} share an edge, the first face F_1 shares an edge with the bottom boundary and the last face F_k shares an edge with the top boundary of the graph.*

Lemma 6.1.10. *A valid face ordering always exists.*

Proof. We use induction to prove a slightly stronger statement: a valid face ordering exists, in which F_1 is incident with a pre-specified edge on the bottom boundary. Clearly the statement is true when the graph has a single face. Suppose it is true for graphs with at most $k - 1$ faces. For a graph with k faces, label F_1 to be the face which shares the specified edge with the bottom boundary. Removing F_1 leaves a planar graph with $k-1$ faces, for which a valid ordering $F_2, ..., F_k$ exists where F_2 shares an edge with F_1 (this edge is on the new bottom boundary). Therefore, $F_1, ..., F_k$ is a valid ordering for the original planar graph. □

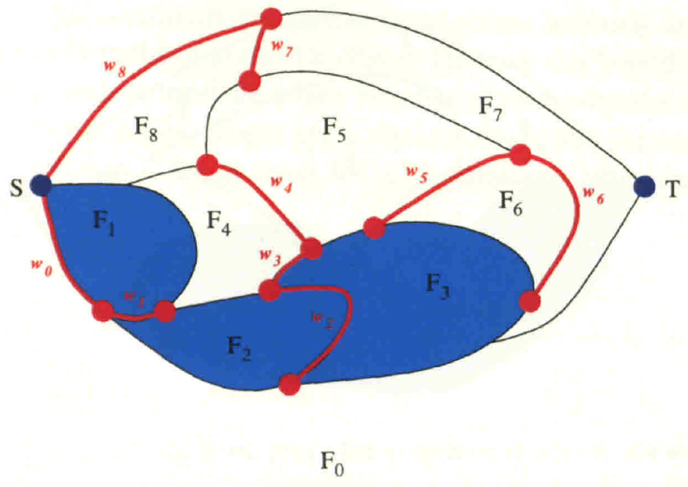


Figure 6.3. The non-zero edges in the graph are $w_0, ..., w_k$ where w_i is the edge between F_i, F_{i+1} in the valid face ordering $\{F_0, F_1, ..., F_k\}$.

There are many valid orderings, any one will do for our construction. One such ordering is shown for the graph in Figure 6.2.

Clearly, every path from S to T is uniquely determined by the set of faces which are enclosed by the path and the bottom boundary. We will show that there are edge weights a_e, b_e such that the segments of the breakpoint function $B(\lambda)$ correspond to the $k+1$ paths determined by F_0, $\{F_1\}$, $\{F_1, F_2\}$,..., $\{F_1, ..., F_k\}$ where F_0 is the empty set or, equivalently for descriptive purposes, the part of the plane under the graph.

Consider any valid face ordering $F_0, F_1, ..., F_k$. Each two consecutive faces in the ordering F_i, F_{i+1} share an edge with parameters a_i, b_i. Set $a_i = \sum_{j=0}^{i} j = \frac{i(i+1)}{2}$, $b_i = k-i$ for $i = 0, 1, ..., k$, and $a_e = b_e = 0$ for all other edges e, as in Figure 6.3. A critical path passing over $\{F_0, F_1, ..., F_r\}$ contains only 0-weight edges and the edge r, therefore the sum of weights along the path is equal to $\sum_{e\in\pi} a_e = a_r = \frac{r(r+1)}{2}$, $\sum_{e\in\pi} b_e = b_r = k - r$.

Any other path passing over $\{F_{i_1}, ..., F_{i_l} = F_r\}$ for $i_1 < ... < i_l = r$ contains at least one non-zero edge r. All other edges have non-negative weights, therefore the sum of edge weights along the path is $\sum_{e\in\pi} a_e \geq a_r = \frac{r(r+1)}{2}$, $\sum_{e\in\pi} b_e \geq b_r = k - r$.

In particular, the path π passing over $\{F_{i_1}, ..., F_r\}$ has a corresponding linear function greater than the linear function of the path passing over $\{F_1, F_2, ..., F_r\}$,

$$\sum_{e\in\pi} a_e + \lambda \sum_{e\in\pi} b_e \geq (1 + 2 + ... + i_r) + \lambda(k - i_r).$$

Therefore, the critical paths which define the linear segments of the breakpoint function $B(\lambda)$, are precisely the paths passing over faces $F_0, \{F_1\}, \{F_1, F_2\}, ..., \{F_1, ..., F_k\}$. Their corresponding linear functions are λk, $1+\lambda(k-1)$, $(1+2)+\lambda(k-2)$, ..., $\frac{k(k+1)}{2} + \lambda * 0$. Any other path lies above the plot of the infimum of linear functions $B(\lambda)$. With this, we have shown that $B(\lambda)$ consists of exactly $k+1$ linear segments and $k+1$ breakpoints. One can easily calculate that the breakpoints are $\lambda = 0, 1, ..., k$.

Theorem 6.1.11. *For every undirected planar graph with k faces, there exist edge weights a_e, b_e which yield $k+1$ breakpoints.*

6.2 Multicriteria optimization

Multicriteria optimization is motivated somewhat similarly by parametric optimization. There are several competing criteria and the goal is to compute the Pareto boundary. The latter consists of all feasible solutions that are not dominated by other solutions, in other words, all solutions for which there is no other solution with lower or equal costs (weights) of all criteria and strictly lower cost of at least one criterion.

We do not offer new results on multicriteria optimization here but rather include this section for clarification on the relationship between it and our stochastic and nonconvex optimization frameworks.

Suppose there are k criteria, specified by weight vectors $(\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_k)$. Geometrically, the Pareto boundary can be best visualized by projecting the feasible set $\mathcal{F}$ onto the subspace of dimension k spanned by these k vectors, and taking the boundary of its dominant, that is its lower-left boundary. Note that this boundary would generally not be convex (for example, with $k = 2$ criteria, the Pareto boundary would look like a staircase).

We do the same in our stochastic framework and general rank-k quasiconcave minimization when the objective function is monotone increasing in $\mathbf{a}_i^T\mathbf{x}$ for all i, except the potential optima here are further restricted to lie on the *convex hull* of the boundary. Thus the local optima of quasi-concave minimization with monotone increasing objectives are a subset of the Pareto boundary. Typically the Pareto boundary is exponentially large, so the focus is on computing ϵ-approximations of it. These consist of a subset $S \subset \mathcal{F}$ of feasible solutions with the property that for every solution $\mathbf{x}$ on the Pareto boundary there is some solution $\mathbf{x}' \in S$ which is at most $(1+\epsilon)$-far from $\mathbf{x}$ in all coordinates:

$$\mathbf{a}_i^T\mathbf{x} \leq \mathbf{a}_i^T\mathbf{x}' \leq (1+\epsilon)\mathbf{a}_i^T\mathbf{x}, \qquad \text{for all criteria } i.$$

An important question in multicriteria optimization is: what is the smallest possible size of an ϵ-approximate Pareto boundary and can it be computed efficiently? Perhaps surprisingly, there is always a succinct approximation, which is polynomial in the problem size (*e.g.,* the size of the ground set n) and in $\frac{1}{\epsilon}$, and exponential in the number of criteria k.

The following theorems from Papadimitriou and Yannakakis [104] show the above and provide a classification for when an ϵ-approximate Pareto boundary is polynomially computable. We include them here with their proofs, as they appear in [104]. Both assume that given a *problem instance* ϑ, if the objective functions (the costs of the criteria $\mathbf{a}_i^T\mathbf{x}$) are positive, then they are between $2^{-p(|\vartheta|)}$ and $2^{p(|\vartheta|)}$ for some polynomial $p(.)$.

Theorem 6.2.1. *[104] For any multicriteria optimization problem and any ϵ there is an ϵ-approximate Pareto boundary consisting of a number of solutions that is polynomial in the size of the problem instance $|\vartheta|$ and $\frac{1}{\epsilon}$ (but exponential in the number of objectives).*

Proof. Consider the k-dimensional space of all criteria. Their values range from $1/2^{p(|\vartheta|)}$ to $2^{p(|\vartheta|)}$ for some polynomial p. Consider now a subdivision of this cube into hyperrectangles, such that, in each dimension, the ratio of the larger to the smaller coordinate is $(1+\epsilon)$. Obviously, there are $O(\frac{(2p(|x|))^k}{\epsilon^k})$ such subdivisions. We define the approximate Pareto boundary by choosing one point of the Pareto boundary in each hyperrectangle that contains such a point. It is easy to see that the resulting set is indeed an ϵ-approximate Pareto boundary. □

Naturally, this is an existence argument and does not tell us whether this approximate Pareto boundary can in fact be computed efficiently. The following theorem provides a characterization for when an efficient computation is possible.

Theorem 6.2.2. *[104] There is an algorithm for constructing an ϵ-approximate Pareto boundary polynomial in the size of the problem instance $|\vartheta|$ and $\frac{1}{\epsilon}$ if and only if the following* GAP *problem can be so solved: Given a feasible solution* $\mathbf{x}$ *and a k-vector $(b_1, ..., b_k)$, either return a solution with $\mathbf{a}_i^T\mathbf{x} \geq b_i$ for all i, or an answer that there is no solution $\mathbf{x}'$ with $\mathbf{a}_i^T\mathbf{x}' \geq (1+\epsilon)b_i$.*

Proof. (If) Suppose that we are given $\mathbf{x}$ and ϵ, and we wish an ϵ-approximate Pareto boundary. Define $\epsilon' = \sqrt{1+\epsilon} - 1 \approx \frac{\epsilon}{2}$ and subdivide the k-space of criteria into hyperrectangles as in the proof of Theorem 6.2.1, using ϵ', and for each corner call the GAP problem. Keep (an undominated subset of) all solutions returned. It is not hard to see that this is an ϵ-approximate Pareto boundary.

(Only if) Conversely, if we have an ϵ-approximate Pareto set, we can solve the GAP problem for any given set of bounds, by looking only at solutions in the set. □

Albeit precise, this classification requires us to know how to solve the auxiliary GAP problem described in Theorem 6.2.2. A more tangible result says that we can compute an approximate Pareto boundary for the multicriteria problem if we have a pseudopolynomial algorithm for the single-criteria problem. This was first used to give an approximate Pareto boundary for the special case of multicriteria shortest paths by Wamburton [128]. The following more general statement is again a theorem from Papadimitriou and Yannakakis [104], and results of similar flavor are also present in Safer, Orlin and Dror [117].

Theorem 6.2.3. *[104] There is an FPTAS for constructing an approximate Pareto boundary for a milticriteria problem, if there is a pseudopolynomial algorithm for the exact version of the single-criterion problem (namely given a problem instance and an integer B, is there a feasible solution with cost exactly B?).*

In particular, an ϵ-approximate Pareto boundary can be computed in time polynomial in the size of the problem instance and in $\frac{1}{\epsilon}$ for the problems:

1. *Shortest path*

2. *Minimum spanning tree*

3. *Matching.*

On the other hand, there is no FPTAS *for constructing an ϵ-approximate Pareto boundary for the* bicriteria $s-t$ Min Cut *problem, unless $P = NP$.*

We can now contrast the results above on multicriteria optimization with our results on stochastic and nonconvex optimization.

If the objective function $f(\mathrm{x}) = \tilde{f}(\mathrm{a}_1^T\mathrm{x}, ..., \mathrm{a}_k^T\mathrm{x})$ in our framework is monotone increasing in $\mathrm{a}_i^T\mathrm{x}$ (so that the local optima lie on the Pareto boundary) and it satisfies a suitable Lipschitz condition or has a bounded gradient (so that x' being $(1+\epsilon)$-close to x implies that their objective function values are also close), then picking the point from the approximate Pareto boundary with smallest objective function value will yield a $(1+\epsilon)$-approximation of our nonconvex problem. However, as we see, some of the natural objective functions already do not fall in the category above (such as the probability tail objective from Section 3.4.1 which has an unbounded gradient), and, if they do, an approximate Pareto boundary may or may not be available even when the underlying linear (single-criterion) problem admits an efficient solution as is the case of $s-t$ Min-Cut.

On the other hand, our results establish the existence of an FPTAS whenever the underlying problem admits an exact efficient solution or an FPTAS (*e.g.*, $s-t$ Min-Cut for which the Pareto boundary cannot be approximated), with the added benefit that the algorithms are independent of the feasible set and thus provide general-purpose solutions.

6.3 Constrained optimization

In this section we compare constrained optimization to our models of stochastic and nonconvex optimization.

For a typical combinatorial optimization problem that seeks a solution x of minimum weight $\mathrm{a}^T\mathrm{x}$, $\{\min \mathrm{a}^T\mathrm{x} \mid \mathrm{x} \in \mathcal{F}\}$, the constrained optimization version seeks a solution of minimum weight which, in addition, does not exceed a given budget with respect to a second cost or weight vector b: $\{\min \mathrm{a}^T\mathrm{x} \mid \mathrm{b}^T\mathrm{x} \leq B,\ \mathrm{x} \in \mathcal{F}\}$. Since knapsack is the easiest special case of a constrained optimization problem, in which the feasible set $\mathcal{F}$ consists of all vertices of the unit hypercube, constrained optimization is in general NP-hard.

Suppose we can get an approximate solution x' to a constrained optimization problem, which satisfies $\mathrm{a}^T\mathrm{x}' \leq \mathrm{a}^T\mathrm{x}^*$ and $\mathrm{b}^T\mathrm{x}' \leq B(1+\epsilon)$, where x^* is the optimal solution to the constrained problem with budget B. Solve the problem for a geometric progression of budgets B and pick the solution x'' with the best objective function value. This will give a fully polynomial-time approximation scheme (FPTAS) to the nonconvex problems (3.4), (3.3), and generally to any problem with a rank-2 objective $f(\mathrm{a}^T\mathrm{x}, \mathrm{b}^T\mathrm{x})$ which is monotone increasing in $\mathrm{a}^T\mathrm{x}$ and in $\mathrm{b}^T\mathrm{x}$, and satisfies a suitable Lipschitz condition. This approach can be used whenever we have the above type of FPTAS to the constrained problem, as is the case for shortest paths [53], for example, or PTAS as in minimum spanning trees [114] (leading to PTAS for the stochastic and nonconvex problems). However, since we do not have a black-box solution to the constrained problem in general, this approach of reducing nonconvex problems to the former does not seem to extend to arbitrary combinatorial problems.

6.4 Stochastic shortest path

In this section we apply our framework and algorithms from Chapters 3, 4 and 6.1 to the shortest path problem, which stands out with its intriguing $n^{O(\log n)}$ complexity. We analyze the optimal stochastic shortest path with respect to the mean-risk, probability tail and value-at-risk objectives as defined in Chapter 3.

Related Work The majority of the related literature on stochastic shortest paths focuses on adaptive algorithms, which compute the next best hop based on information about realized edge lengths so far [10], [111], [16], [103], [37], [93]. Most of the adaptive formulations focus on minimizing expected path length; few consider minimizing a non-linear function of the length and settle for heuristic algorithms [37].

The most closely related nonadaptive formulation to our model is that of Loui [84]. Loui considers a general utility function of path length which is monotone and nondecreasing, and proves that the expected utility becomes separable into the edge lengths only when the utility function is linear or exponential. In that case the path that maximizes expected utility can be found via traditional shortest path algorithms. For general utility functions he gives an algorithm based on an enumeration of paths, with a very large running time $O(n^n)$. In a consequent paper, Mirchandani and Soroush give exponential algorithms and heuristics for quadratic utility functions [94]. For non-monotone utility functions Nikolova, Brand and Karger [98] give hardness results and pseudopolynomial algorithms, described in Chapter 2 of this thesis. For a separate model on bicriteria shortest paths with monotone objective, Ackerman *et al.* [3] give different average and smoothed analyses.

6.4.1 Exact algorithms

We follow the same notation as in Chapter 3. For shortest paths, we have a graph $G = (V, E)$, with $|V| = n$ nodes and $|E| = m$ edges. We are given a source S and destination T and would like to find ST-paths. Each path is represented by its incidence vector $\mathbf{x} \in \mathbb{R}^m$ whose coordinate x_i is 1 if edge i is present in the path and 0 otherwise. Additionally, the vector of edge weights is $\mathbf{W} \in \mathbb{R}^m$, whose coordinates are random variables with means $\boldsymbol{\mu}$ and variances $\boldsymbol{\tau} \in \mathbb{R}^m$.

Recall that in the stochastic optimization framework from Chapter 3, the optimal solutions are extreme points on the dominant (*i.e.*, the lower-left boundary) of the projected path polytope onto the mean-variance plane $span(\boldsymbol{\mu}, \boldsymbol{\tau})$. On the other hand, the number of such extreme points equals the number of breakpoints of the parametric shortest path problem, $n^{1+\log n}$. Since all such extreme points can be found with at most twice as many calls to a deterministic shortest path algorithm, this gives rise to exact algorithms with running time $n^{O(\log n)}$.

Theorem 6.4.1. *There is an exact algorithm with running time $n^{O(\log n)}$ for solving the stochastic shortest path problem under the:*

- *mean-risk objective $\boldsymbol{\mu}^T\mathbf{x} + c\sqrt{\boldsymbol{\tau}^T\mathbf{x}}$ with arbitrary distributions;*

- *value-at-risk objective;*

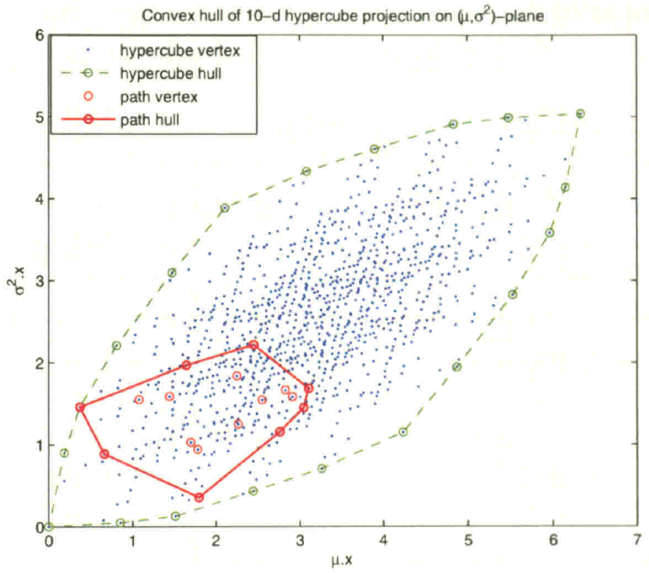


Figure 6.4. Projection of the unit hypercube (representing all edge subsets) and the path polytope onto the (μ, σ^2)-plane.

- *probability tail objective* $\mathbf{Pr}(\mathbf{W}^T\mathbf{x} \leq t)$ *with normal distributions, assuming* $\boldsymbol{\mu}^T\mathbf{x} \leq t$ *for at least one path.*

We mention that by the results in Chapter 3, all the problems above admit FPTAS.

Corollary 6.4.2. *There are fully-polynomial approximation schemes for all stochastic shortest path variants above.*

For the probability tail objective, when the departure time is closer to the deadline, so that any shortest path has mean greater than t, the optimal value of the program (3.4) will be negative and the objective function is decreasing in the mean of the path and increasing in the variance. In this case the solution to the fractional version of the program will be on the upper-left boundary of the shadow, between the leftmost extreme point of lowest mean and the uppermost extreme point of highest variance (See Figure 6.4). Since finding the simple path with highest variance is strongly NP-hard [73], we might not expect to find a good polynomial-time approximation. In addition, the objective function in (3.4) is no longer quasiconvex so that even if we could find the fractional solution, it is not guaranteed to coincide with the integer solution to the problem.

We propose a pseudopolynomial algorithm that may possibly include non-simple paths, which is polynomial in the number of nodes n and the maximum edge variance τ_{max}. The algorithm constructs a dynamic programming table as follows: For each possible value $s = 0, 1, ..., n\tau_{max}$ of a path variance, and for each node v, compute v's predecessor on a path from the source to node v of variance s and smallest mean. At the end, we would have $n\tau_{max}$ paths from the source to the destination, one for each possible variance $s \in \{0, 1, ..., n\tau_{max}\}$ and smallest mean for that variance value. More precisely, for each node v and variance s, we solve the suboptimality equations

$$\Phi(v,s) = \min_{v' \sim v} \left[\Phi(v', s - \tau_{v'v}) + \mu_{v'v}\right],$$

$$\pi(v,s) = arg\min_{v' \sim v} \left[\Phi(v', s - \tau_{v'v}) + \mu_{v'v}\right],$$

where $\Phi(v,s)$ is the mean of the smallest-mean path from the source to v of variance exactly s; $\pi(v,s)$ is the predecessor of v on that path, and for a neighbor v' of node v, $\mu_{v'v}, \tau_{v'v}$ are the mean and variance of edge $v'v$. Finally, we find the optimal ST-path by computing the objective function value $\frac{t\text{−path mean}}{\sqrt{\text{path variance}}}$ for each partially optimized path of variance s and select the path with smallest objective value. Thus, the algorithm's total running time is $O(nm\tau_{max})$.

Theorem 6.4.3. *For general deadline t, the stochastic shortest path problem with probability tail objective and normal edge weight distributions can be solved in time $O(nm\tau_{max})$ where τ_{max} is the maximum variance of an edge.*

6.4.2 Extensions to other distributions

To contrast the high sensitivity of the problem complexity on the distributional assumptions, we now consider the stochastic shortest path problem with probability tail objective under distributions other than the normal.

Poisson and additive stochastic dominant distributions—exact solution

The probability distribution $\mathcal{D}(\lambda)$ is called *additive* if the sum of two independent random variables with distributions $\mathcal{D}(\lambda_1)$ and $\mathcal{D}(\lambda_2)$ is another random variable with the same distribution and parameter equal to the sum $\mathcal{D}(\lambda_1 + \lambda_2)$. With a slight abuse of notation we use $\mathcal{D}(\lambda)$ to also denote a random variable with this distribution. Assume, in addition, that the distribution $\mathcal{D}$ satisfies *stochastic dominance*, that is $\mathbf{Pr}(\mathcal{D}(\lambda_1) \leq t) \geq \mathbf{Pr}(\mathcal{D}(\lambda_2) \leq t)$ whenever $\lambda_1 \leq \lambda_2$. Examples of such distributions are Poisson and $gamma(a,b)$ with constant parameter b.

Suppose the random length of edge i is $W_i \sim \mathcal{D}(\lambda_i)$. Now, despite the non-separable objective function, the form of distribution makes the problem separable:

$$\mathbf{Pr}\left(\sum_{i\in\pi} W_i \leq t\right) = \mathbf{Pr}\left(\sum_{i\in\pi} \mathcal{D}(\lambda_i) \leq t\right) = \mathbf{Pr}\left(\mathcal{D}(\sum_{i\in\pi}\lambda_i) \leq t\right) \geq \mathbf{Pr}\left(\mathcal{D}(\lambda') \leq t\right),$$

where the last inequality follows from the stochastic dominance property of the distribution for all $\lambda' \geq \sum_{i\in\pi}\lambda_i$. With this, the optimal path is the one that has the smallest sum of distribution parameters along its links and can be found exactly with a deterministic shortest path algorithm.

Exponential PTAS and Bernoulli QPTAS

Unlike the Poisson, the exponential distribution is not additive and we cannot write a simple closed form expression for the objective function. We propose a polynomial-time approximation scheme based on dynamic programming over a discretization of the distribution parameter space. More precisely, we give a bi-criteria approximation, which is given a lateness tolerance p and for $\epsilon > 0$ finds an ST-path π satisfying

$$\mathbf{Pr}\Big(\sum_{i\in\pi} W_i < t(1+\epsilon)\Big) > 1 - \epsilon p.$$

Theorem 6.4.4. *An approximately optimal path π with $\mathbf{Pr}[\sum_{i\in\pi} W_i > (1+\epsilon)] \le p(1+\epsilon)$ can be computed in time $O(n^4 \log n) O\Big(\frac{\log(1/\epsilon)}{\epsilon^4} \log \frac{1}{\epsilon p}\Big)\gamma^{O(\frac{1}{\epsilon}\log 1/\epsilon)}$.*

Proof. Note that dividing all edge lengths by t would be equivalent to the problem of finding a path with $\mathbf{Pr}\Big(\sum_{i\in\pi} W_i/t < (1+\epsilon)\Big) > 1 - \epsilon p$, and we discuss this problem instead.

This problem is similar to the stochastic knapsack formulation considered by Goel and Indyk [52]. There, the authors aim to find a set of (stochastic size) items of maximum value, such that the probability of exceeding the knapsack capacity is at most p. Our problem is different since, instead of maximizing values, we have an additional constraint that the set of edges we choose have to form a path, but a similar discretization of the distribution space will help us as well.

Divide the edges into small and large where small edges have mean below a threshold α and large edges have mean between α and β, where in the case of exponential distributions $\alpha = \frac{\epsilon^3}{\log(1/\epsilon)\log(1/p)}$ and $\beta = \frac{1}{\log(1/p)}$. (Large edges with greater mean would alone exceed the allowed lateness probability and we ignore them.) Round down the means of the large edges to the largest value $\frac{1/\log(1/p)}{(1+\epsilon)^i}$ for $i = 0, 1, 2, ...$

Next, we fill in a dynamic programming table DP in which rows correspond to each possible set type L of large edges together with each possible level S of small edges, where $S = j * \frac{\epsilon}{n}$ with $j \in \{0, 1, ..., \lceil \frac{n}{\epsilon} \rceil\}$. The columns correspond to the vertices. The cell $DP(\{L, S\}, v)$ is empty if there is no path from v to the destination, which consists of an L-set of large edges and small edges of total mean rounded up to S, with a rounding interval of $\frac{\epsilon}{n}$. Otherwise, the entry contains v's successor node on the path and the exact value of the total mean of the small edges on the path. In the case of more than one such path, we only keep a record of the path with lowest total mean of small edges.

For each row, we iteratively go through every node v in the graph and update the entry $DP(\{L, S\}, v)$. In particular, we consider each neighbor v' of v for which the entry $DP(\{L, S\} - \mu_{vv'}, v')$ is non-empty and results in a path with lower total mean of small edges than already recorded in $DP(\{L, S\}, v)$. With a slight abuse of notation, if $\mu_{vv'}$ is large, $\{L, S\} - \mu_{vv'}$ is defined as $\{L', S\}$ where L' is the same as L with the number of $\mu_{vv'}$-type edges reduced by one. Similarly, if $\mu_{vv'}$ is small, then $\{L, S\} - \mu_{vv'} = \{L, \lceil S - \mu_{vv'} \rceil\}$, where the total length of small edges is rounded up to the closest level $j * \frac{\epsilon}{m}$. Note that in this case it is possible that after the rounding $S' = S$. Thus, sometimes we would be filling a table cell by relying on another cell from the same row, which has not yet been filled. To complete filling in the row correctly, we would need to repeat the same row iteration up to n times.

When the whole DP table has been filled, we go through all entries in the first column of the table, corresponding to all possible solution paths from the source to the destination. For each, we calculate the probability that the sum of the edge length variables on the corresponding path exceeds the lateness threshold $p(1+\epsilon)$ and we pick the path with the smallest probability of being late.

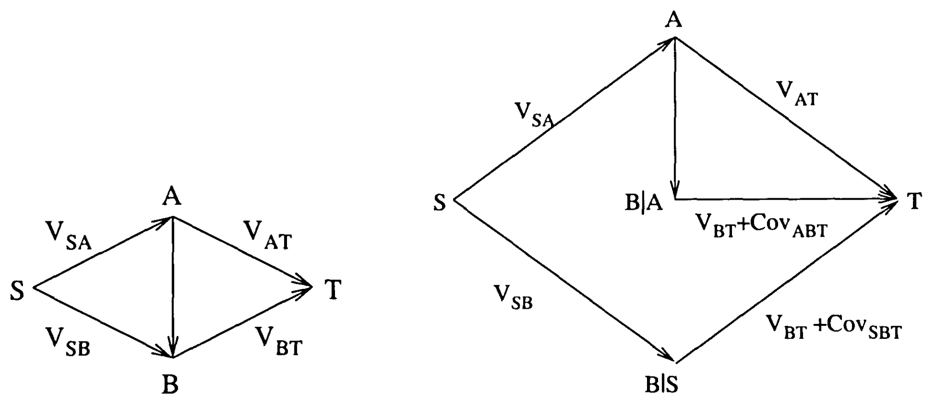


Figure 6.5. Solution with correlations between adjacent edges.

Computing the probability of being late on a path is the same as in computing the probability that the selected items in the stochastic knapsack exceed the knapsack capacity [52]: by discretizing the edge lengths we can compute the probability in time $O(\frac{n^3}{\epsilon}\log\frac{n}{\epsilon p})$. The overall running time of the algorithm depends on the size of the discretized distribution space (as in Goel and Indyk [52]) and the construction of the dynamic programming table above and is omitted. □

A similar discretization of the state space yields a quasi-polynomial approximation scheme for the case of Bernoulli distributions.

6.4.3 Correlations

In practice, assuming independent distributions for neighboring edges in a graph is often unrealistic; for example, in a shortest paths problem, an accident in one edge would increase congestion in the edges that follow it. Nevertheless, the algorithms that we have developed under the assumption of independent edge distributions would still apply in this case, after an appropriate graph transformation, which was suggested to us by Alexander Hartemink [66]. We describe this transformation below, which works for arbitrary distributions.

Suppose there are pairwise correlations between adjacent edges (the first one incoming and the second outgoing from their common node). Consider the following graph transformation: For every node B with incoming edges (A, B) in the original graph G, create nodes $B|A$ in the new graph G'. An edge (A, B) in G yields edges $(A|X, B|A)$ in G', for all nodes X that precede node A in G. Denote the covariance between edges (A, B) and (B, T) in G by Cov_{ABT}, and their variances by V_{AB} and V_{BT} respectively. Then, in the transformed graph G', define the variance of edge $(B|A, T|B)$ by $V_{BT} + Cov_{ABT}$ as in Figure 6.5. Notice that these definitions of variance and, generally, of conditional distributions on the edges in the transformed graph, decouple the correlations and the edge distributions are now independent. We can run our existing algorithms on G' and in this way solve the problems for correlated edges in the original graph G. We can apply this method of decoupling correlated edges for not just correlations between two neighboring edges, but up to a constant number of consecutive edges (in order to maintain polynomial size for the transformed graph G').

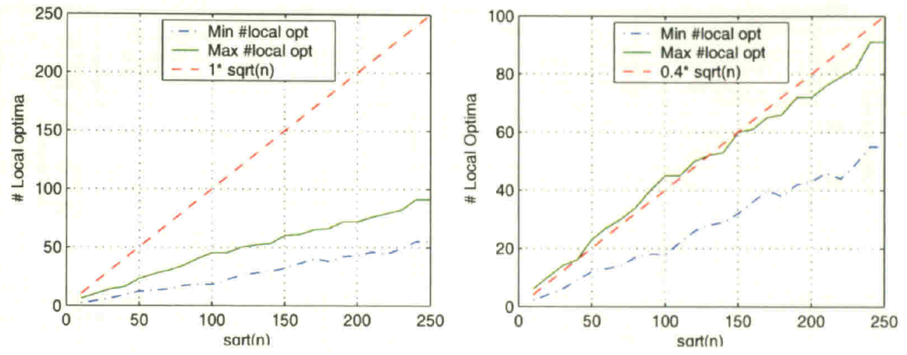


Figure 6.6. Number of local optima (extreme point paths) in grids of size 10×10 to 250×250, with edge mean length $\mu_e \leftarrow Uniform[0, 1]$ and variance $\tau_e \leftarrow Uniform[0, \mu_e]$.

6.4.4 Experimental results

Performance of exact algorithm

In this section, we investigate the practical performance of exact and approximation algorithms for the stochastic routing problem in which the goal is to find the route minimizing a positive linear combination of the route mean and standard deviation, $\alpha\boldsymbol{\mu}^T\mathbf{x} + (1-\alpha)\sqrt{\boldsymbol{\tau}^T\mathbf{x}}$. The latter is a nonconvex integer optimization problem, for which no efficient algorithms are known. In summary, our experimental results show that the exact algorithm, which is based on enumerating all paths that are potential local optima (extreme points of the feasible set), has surprisingly good running time performance $O(\sqrt{n})R$ on networks of practical interest compared to its predicted theoretical worst-case performance $n^{O(\log(n))}R$, where R is the running time of any deterministic shortest path algorithm of the user's choice. We also show that the approximation algorithm, which appropriately selects a small subset of the potentially optimal paths, has very high performance, using a very small constant number of deterministic shortest path iterations and returning a solution that has a 99.99% accuracy. Therefore approximation algorithms of this type are a very promising practical approach.

The exact algorithm for solving the problem can be thought of as a search algorithm on the values of the parameter β for the oracle minimizing the *linear* objective $\beta\boldsymbol{\mu}^T\mathbf{x} + (1-\beta)\boldsymbol{\tau}^T\mathbf{x}$. For a given β and a path P with incidence vector $\mathbf{x}$, the objective can be separated as a sum of edge weights $\sum_{i\in P}(\beta\mu_i + (1-\beta)\tau_i)$ (note, the variance of the path P equals the sum of variances of the edges along the path, by the independence of the edge length distributions). Thus, finding the ST-route that minimizes the linear objective can be done with any deterministic shortest path algorithm such as Dijkstra, Bellman-Ford, etc. [26] with respect to edge weights $(\beta\mu_i + (1-\beta)\tau_i)$.

Both the exact and approximation algorithms we consider will consist of a number of calls to a deterministic shortest path algorithm of the user's choice for appropriately chosen values β: this makes our approach very flexible since different implementations of shortest paths algorithms are more efficient for different types of networks and one can take advantage of the most efficient implementations available. We therefore characterize the running time performance in terms of the number of such calls or iterations to an underlying shortest path algorithm.

In the worst case the number of extreme point paths can be $n^{1+\log n}$ [99]—that is superpoly-

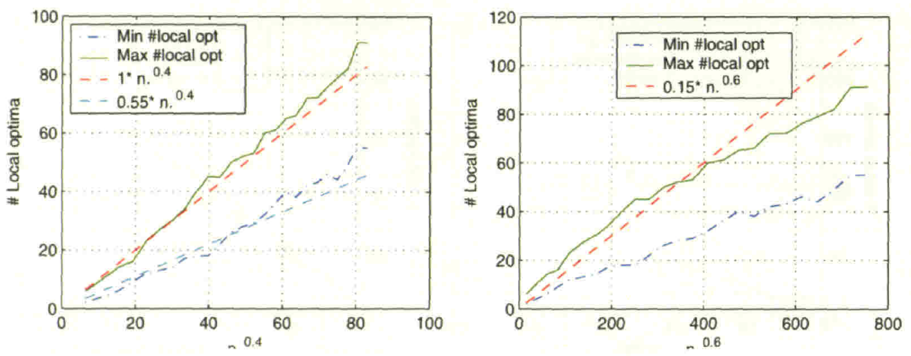


Figure 6.7. Number of local optima in grids with n nodes vs $n^{0.4}$ (left) and $n^{0.6}$ (right).

nomial (albeit subexponential) and too large to yield an efficient algorithm. However, in many networks and mean-variance values of interest, this number seems to be much lower, implying that the exact algorithm may have a good performance in practice.

We therefore investigate the performance of the algorithm in a class of networks that are the standard first test set for indicating performance on realistic traffic networks. We consider grid networks of size ranging from 10×10 (100 nodes) to 250×250 ($62,500$ nodes) in additive increments of 10×10. For each network size type $10z \times 10z$ (where $z = 1, ..., 25$), we run 100 instances of the stochastic routing problem. In a single instance, we generate the mean values uniformly at random from $[0, 1]$, and the variance values uniformly at random from the interval $[0, mean]$ for a corresponding edge with an already generated $mean$ value. (By scaling all edge means if necessary, we can assume without loss of generality that the maximum mean has value 1.) Out of these 100 simulations per network size, we record the minimum and the maximum number of extreme points and plot them against the square root of the network size (*i.e.,* the square root of the number of nodes in the network). The resulting plots are shown in figure 6.6.

To put these empirical results in context: The maximum number of extreme points on a network with $10,000$ nodes found from the simulations, is $k = 45$ (meaning the exact algorithm consisted of only $2k = 90$ iterations of a deterministic shortest path algorithm to find the *optimal* stochastic route) as opposed to the predicted worst case value of $10,000^{1+\log 10,000} \approx 10^{57}$! Similarly, the highest number of extreme points found in graphs of $40,000$ and $62,500$ nodes is $k = 75$ and $k = 92$ respectively as opposed to the theoretical worst-case values of $40,000^{1+\log 40,000} \approx 10^{75}$ and $62,500^{1+\log 62,500} \approx 10^{81}$. In other words, despite the pessimistic theoretical worst-case bound of the exact stochastic routing algorithm, it has good performance in practice that is orders of magnitude smaller.

In figure 6.6, we have fitted the maximum number of extreme points to a linear function of $\sqrt{n}$: the extreme points for a graph with n nodes are always strictly less than $\sqrt{n}$ and asymptotically less than $0.4\sqrt{n}$. To ensure that $\sqrt{n}$ is the right function to bound the asymptotics, we have also plotted the number of extreme points with respect to $n^{0.4}$ and $n^{0.6}$, in figure 6.7. The latter two plots confirm that the rate of growth of extreme points is faster than $\theta(n^{0.4})$ and slower than $\theta(n^{0.6})$.

On the basis of these empirical results, we conclude that a strict upper bound on the number of iterations of the exact algorithm for any grid graph with n nodes is $\sqrt{n}$. We leave as an intriguing

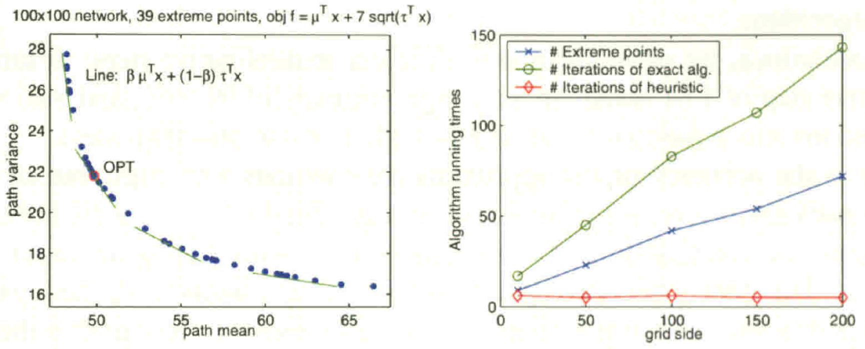


Figure 6.8. *(left)* Plot of all extreme point paths of a $10,000$-node network. The optimal path is marked with a circle and labelled 'OPT'. The approximation algorithm minimizes a small number of linear functions as shown, each yielding one extreme point. It then outputs the extreme point (path) with smallest objective function value. *(right)* Number of iterations in the approximation versus the exact algorithm, in 10x10 up to 200x200 grid networks. Recall that the number of iterations to find all extreme points in the exact algorithm is two times the number of extreme points, where each iteration is a call to a deterministic shortest path algorithm of the user's choice.

open problem to give a theoretical proof of this performance bound.

Performance of approximation algorithms

In this section we evaluate the approximation algorithms from Chapter 3, which dramatically reduce the running time of the exact algorithm above. Instead of enumerating all extreme points, we select a very small subset of them, leading to high practical performance of the algorithm.

Remarkably, the big reduction of extreme points does not lead to a big sacrifice in the quality of the resulting solution. Our experiments show that even on large networks of $40,000$ nodes, the approximation algorithm examines only 3 to 6 extreme point paths (compare to the experimental 75 and the theoretical worst bound of $40,000^{1+\log 40,000} \approx 10^{75}$ points of the exact algorithm above), and in all our simulations the value of the solution is within a multiplicative factor of $0.0001 = 0.01\%$ of the optimum.

The approximation algorithm tests an appropriate geometric progression of values β in the linear objective function $\beta\boldsymbol{\mu}^T\mathbf{x} + (1-\beta)\boldsymbol{\tau}^T\mathbf{x}$ and selects the best of the resulting small set of paths. We include an illustration of the algorithm in figure 6.8*(left)*. This figure plots all extreme point-paths for a $10,000$-node network (with mean and variance values of its edges generated as explained in the previous section). Each point in the figure corresponds to one path in the network with mean equal to the x-coordinate of the point, and variance equal to the y-coordinate of the point. In this mean-variance plot, the optimum happens to be a path with mean 49.83 and variance 21.77 (marked by a circle in the figure). As depicted on the figure, the optimal path minimizes $\beta\boldsymbol{\mu}^T\mathbf{x} + (1-\beta)\boldsymbol{\tau}^T\mathbf{x}$ (this corresponds to a line with slope $b = -\frac{\beta}{1-\beta}$) for some range of parameter values β or equivalently for some range of slopes.

If we could guess the optimal slope, then we would find the optimal route with a single iteration of a deterministic shortest path algorithm with respect to edge weights $\beta\mu_i + (1-\beta)\tau_i$. Instead, we test a geometric progression of slopes with a multiplicative step a. The smaller the step a, the more linear objectives we end up testing, which increases the accuracy but also increases the running time of the algorithm.

In our simulations, we experiment with different multiplicative steps. It turns out that using a multiplicative step of 1.01 results in very high accuracy of 99.99%, and also very few extreme point tests and iterations (up to 6 for all graphs with 2, 500 to 40, 000 nodes).

Not only is the accuracy of the approximative solutions very high, but in over 85% of the instances for each graph size, the approximation algorithm finds exactly the optimal route.

We compare the running time of the approximation with that of the exact algorithm in figure 6.8*(right)*. This plot gives the highest number of deterministic shortest path iterations that each algorithm has run over 100 simulations per network size. Recall that the exact algorithm needs to run two times as many iterations as the total number of extreme points (paths minimizing $\beta\boldsymbol{\mu}^T\mathbf{x} + (1-\beta)\boldsymbol{\tau}^T\mathbf{x}$ for some β). The plot shows that the high performance and accuracy of the approximation algorithm makes it a very practical and promising approach for the stochastic routing problem.

Bibliographic notes. Part of the material presented in Chapters 2, 4 and 5 has appeared in joint publications with Matthew Brand, David Karger, Jonathan Kelner and Michael Mitzenmacher [98, 99, 78].

Part II

Measuring Uncertainty through Economics: *Mechanisms and Algorithms for Prediction Markets*

CHAPTER 7

Introduction to prediction markets

A key component in models involving uncertainty is the random variables distribution, often assumed to be known or having known parameters. In this section we provide new mechanisms and algorithms for an increasingly common and accurate method of estimating uncertainty and its parameters: prediction markets, also known as information or decision markets.

Prediction markets are structured similarly to financial markets, except the main purpose is to aggregate traders' information so as to predict the outcome of an event. It has long been observed that trading reveals information. Buying or selling a financial security in effect is a wager on the security's value. For example, buying a stock is a bet that the stock's value is greater than its current price. Each trader evaluates his expected profit to decide the quantity to buy or sell according to his own information and subjective probability assessment. The collective interaction of all bets leads to an equilibrium that reflects an aggregation of all traders' information and beliefs. In practice, this aggregate market assessment of the security's value is often more accurate than other forecasts relying on experts, polls, or statistical inference [108, 109, 44, 8, 107].

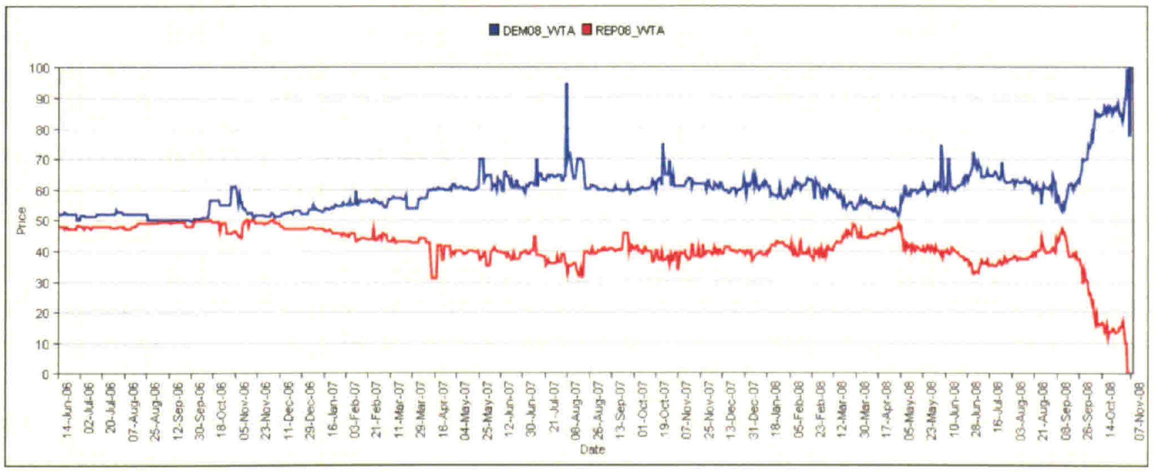


Figure 7.1. Price graph for the U.S. presidential elections winner takes all market from June 1, 2006 until November 7, 2008 in the Iowa Electronic Markets.

Consider buying a security at price fifty-two cents, that pays \$1 if and only if a Democrat wins the US Presidential election. The transaction is a commitment to accept a fifty-two cent loss if a Democrat does not win in return for a forty-eight cent profit if a Democrat does win. In this case of an *outcome-contingent security*, the price—the market's value of the security—

corresponds directly to the estimated probability of the outcome of the event. Figure 7.1 presents the corresponding price graph for the historic 2008 presidential election of Barack Obama vs John McCain. Traders bet on which party would win the 2008 presidential elections. The market started operation on June 1, 2006 and, as the graph shows, the market prices accurately predicted, more than two years prior to the elections, that the democratic candidate (unknown at that time) would win.

Trading prices for the presidential election securities in Figure 7.1 are taken from the Iowa Electronic Markets, one of the oldest operating prediction markets and one of few such markets operating with real money. It is a non-for-profit market run for educational and research purposes at the University of Iowa Tippie College of Business. Compared to traditional opinion polls on past U.S. presidential elections, the Iowa Electronic Market was more accurate 451 out of 596 times [8].

Another notable example of prediction markets—the Hollywood Stock Exchange, established in 1996, correctly predicted 32 of 2006's 39 big-category Oscar nominees and 7 out of 8 top category winners.

Controversy temporarily stalled prediction markets research when, in the summer of 2003, a market proposed for analyzing and predicting political and economic stability in the Middle East gained the unfortunate misnomer "the terrorist futures market" and prediction markets research funding was cancelled [110]. The notorious example which led to this misnomer was proposing to predict terrorist attacks or assassinations: fear arose that this would create the wrong incentives and lead to a self-fulfilled prophesy. One of the market designers, Hanson, as well as other leading economists refuted those fears in follow-up articles [65].

So what makes prediction markets so accurate and at the same time such a flourishing area of open research questions?

A lot of what we know about prediction markets accuracy and results stems from empirical data about existing prediction markets such as the Iowa Electronic Markets, TradeSports, the Hollywood Stock Exchange, etc. Since the market designs and trading behavior are very complex to capture in realistic theoretical models, the field of prediction markets leaves open fundamental questions such as what are optimal prediction market designs and what is the optimal strategic behavior.

In Part II we present strategic analysis of prediction markets that gives a unified theoretical approach to understanding their properties, as well as optimal trading behavior. We then propose and analyze several new market designs for handling exponentially many permutation or ranking-type outcomes.

Prediction market designs can be divided into two main categories. In one category (Chapter 9) they operate as an *auction* in which orders on the different securities are matched risklessly by a book-keeper. This market structure leads to a complex price formation process in which it may be difficult to infer resulting prices and the corresponding probabilities of the outcomes. Another challenge is that we may not be able to find matching traders who agree on a price and trade would not occur—the so-called "thin markets" problem. This thesis presents new designs for handling exponentially large outcome spaces resulting from permutation or ranking-type outcomes where the event one is trying to predict is the result of a tournament or a competition of n candidates. Our work is similar in spirit to that of Fortnow *et al.* [45] who instead consider prediction market designs for events whose outcomes can be expressed as logical formulas, and prove hardness results for the auctioneer problem in that setting. With a large number of outcomes it is impractical to have a security representing each possible outcome: we need to restrict the number and types of securities, which means restricting the type of information or beliefs the traders can express. For example, if we only had n securities for the n possible winners in the tournament, we would lose as a participant a trader who does not have information on the winner, but instead has some belief about the ranking of a subset of the candidates (*e.g.,* candidate A would do better than candidate B): this trader would not be able to express such belief by trading with only these n securities. On the other hand, a higher number of securities and higher expressivity may lead to computationally intractable problems of finding matching traders who can trade.

Somewhat surprisingly, we show that in this perceived trade-off between expressivity and tractability, these two goals for practical prediction market design are not necessarily conflicting. With one family of securities, in which each security corresponds to a pair of ranked candidates, we both have a limited expressivity of only $O(n^2)$ possible securities (one for each ordered pair), and the problem of identifying matching traders is NP-hard. With another family of securities where traders can bid on which subset of candidates would place in a given position or which subset of positions a candidate would end up in, we have exponentially many possible securities and at the same time we can solve the auctioneer problem—to identify matching traders—in polynomial time. Curiously the economic analysis of the auctioneer's revenue maximization and matching problems is based on classical results from combinatorial optimization such as the NP-hardness of the minimum feedback arc problem and the polynomial-time solvability maximum matching.

In the other category of prediction markets (considered in Chapter 8) the challenges of thin markets and complex price formation are alleviated by having a deterministic price-setting rule based on trading volume, and a centralized *market maker* who is always willing to trade at the given price. A drawback in this type of market is that the market maker may incur a loss by fulfilling all orders at the current price. Nevertheless, such loss is not necessarily a negative, as long as it is bounded, when viewed as a subsidy or price that the market maker incurs in order to aggregate and acquire information. Despite the growing popularity and implementation of prediction market designs with a market maker, their strategic and theoretical properties are not fully understood. For example, the dynamic parimutuel market proposed by Pennock [106] was implemented in the Yahoo! Buzz Market and a previous version crashed after traders found and exploited arbitrage opportunities [86], calling for a change in the market design. In this chapter, we propose an abstract prediction market, and show how to use it as a tool for analyzing the strategic properties of existing market designs, as well as for guiding future designs. In particular, we prove that our prediction market is strategically equivalent to existing market mechanisms and thus its analysis automatically transfers to these mechanisms. The benefit of using our abstract market model is that it has an attractive geometric visualization which makes the analysis and strategy computation significantly easier than in the market designs operating in practice.

CHAPTER 8

Prediction markets with a market maker

In markets with a market maker, there is a deterministic price setting rule based on trading volume. The market maker guarantees to fulfill all orders at a given price. The existing market designs differ in price rules, as well as the rules on liquidating the winning securities at the end. In the following subsections we give an overview of two such prediction markets: the Market Scoring Rules and the Dynamic Parimutuel Markets. The incentives and strategic behavior in these markets have not been well understood. We proceed to give a unified strategic analysis via a new simplified prediction market design which we call the *Projection Market.*

8.1 Dynamic parimutuel markets

The dynamic parimutuel market (DPM) was introduced by Pennock [106] as an information market structure that encourages informed traders to trade early, has guaranteed liquidity, and requires a bounded subsidy. This market structure was used in the Yahoo! Buzz market [86].

Let $A_1, ..., A_n$ denote the n mutually exclusive and exhaustive outcomes of an event. In a DPM, users can place bets on any outcome A_i at any time, by buying a *share* in the appropriate outcome. The price of a share is variable, determined by the total amount of money in the market and the number of shares currently outstanding. Further, existing shares can be sold at the current price. After it is determined which event really happens, the shares are liquidated for cash. In the "total-money-redistributed" variant of DPM, which is the variant used in the Yahoo! market, the total money is divided equally among the shares of the winning outcome; shares of the losing outcome are worthless. Note that the payoffs are undefined if the event has zero outstanding shares; the DPM rules should preclude this possibility.

We use the following notation: Let x_i be the number of outstanding shares of A_i (totaled over all traders), for $i = 1, ..., n$. Let M denote the total money currently in the market. Let c_i denote the price of shares in A_i.

Two price-rules were initially proposed for determining the price of a share: the *money-ratio* and the *share-ratio* principle. According to the money-ratio, the ratio of prices of two securities is equal to the ratio of the total money spent on these securities:

$$\frac{c_i}{c_j} = \frac{M_i}{M_j}.$$

Similarly according to the share-ratio, the ratio of prices of two securities is equal to the ratio of the number of outstanding shares in these securities:

$$\frac{c_i}{c_j} = \frac{x_i}{x_j}. \tag{8.1}$$

Initially, the Yahoo! Buzz market was implemented using the money-ratio price rule. However, in that market traders were able to find and exploit arbitrage opportunities, causing a market collapse [86]. The price rule was then changed to the share-ratio principle.

The form of the prices in the DPM can be fully determined by stipulating that, for any given value of M and $x_1, ..., x_n$, there must be some probability p_i such that, if a trader believes that p_i is the probability that A_i will occur and the market will liquidate in the current state, she cannot expect to profit from either buying or selling either share. This gives us

$$c_i \ = \ p_i\Big(\frac{M}{x_i}\Big).$$

Since $\sum_{i=1}^{n} p_i = 1$, we have:

$$\sum_{i=1}^{n} x_i c_i = M. \tag{8.2}$$

On the other hand, Equation (8.1) from the share-ratio principle leads to the following equivalent reformulation:

$$\begin{aligned} & \frac{c_i}{x_i} x_j = c_j \\ \Leftrightarrow \quad & \frac{c_i}{x_i} x_j^2 = c_j x_j \\ \Leftrightarrow \quad & \frac{c_i}{x_i} \sum_j x_j^2 = \sum_j c_j x_j. \end{aligned}$$

Finally, substituting $\sum_j c_j x_j = M$ from Equation (8.2) into the last equation above gives the form of instantaneous prices under the share-ratio DPM:

$$c_i \ = \ x_i \frac{M}{|\mathbf{x}|^2} \qquad \forall i, \tag{8.3}$$

where $|\mathbf{x}| = \sqrt{x_1^2 + ... + x_n^2}$ is the Euclidian norm of vector $\mathbf{x}$.

Theorem 8.1.1. *The cost of the move from* $\mathbf{x}$ *to* $\mathbf{x}'$ *is*

$$M' - M = M_0(|\mathbf{x}'| - |\mathbf{x}|)$$

for some constant M_0.

Proof. Consider the function $G(\mathbf{x}) = M_0[\sqrt{x_1^2 + ... + x_n^2}]$. The function G is differentiable for all $x_i \neq 0$, and its partial derivatives are:

$$\frac{\partial G}{\partial x_i} = M_0[\frac{x_i}{\sqrt{x_1^2 + ... + x_n^2}}] = x_i \frac{G(\mathbf{x})}{|\mathbf{x}|^2}$$

Now, compare these equations to the prices in the DPM, and observe that, as a trader buys or sells in the DPM, the instantaneous price is the derivative of the money. It follows that, if at any point of time the DPM is in a state (M, x) such that $M = G(\mathrm{x})$, then, at all subsequent points of time, the state (M', x') of the DPM will satisfy $M' = G(\mathrm{x}')$. Finally, note that we can pick the constant M_0 such that the equation is satisfied for the initial state of the DPM, and hence, it will always be satisfied. □

An interesting corollary of Theorem 8.1.1 is that the instantaneous prices for all assets sum to more than 1.

8.2 Market scoring rules

The Market scoring rule (MSR) was introduced by Hanson [64]. It is based on the concept of a *proper scoring rule*, a technique which rewards forecasters to give their best prediction. Hanson's innovation was to turn the scoring rules into instruments that can be traded, thereby providing traders who have new information an incentive to trade. One positive effect of this design is that a single trader would still have incentive to trade, which is equivalent to updating the scoring rule report to reflect her information, thereby eliminating the problem of thin markets and illiquidity.

Proper scoring rules are tools used to reward forecasters who predict the probability distribution of an event. In the setting of n exhaustive, mutually exclusive outcomes $A_1, ..., A_n$, proper scoring rules are defined as follow:. Suppose the forecaster predicts that the probabilities of the outcomes are $\mathrm{r} = (r_1, ..., r_n)$, with $\sum_i r_i = 1$. The scoring rule is specified by functions $s_i(\mathrm{r})$ for all i, which are applied as follows: If the outcome A_i occurs, the forecaster is paid $s_i(\mathrm{r})$. The key property that a proper scoring rule satisfies is that the *expected* payment is maximized when the report is identical to the true probability distribution. Denote the score functions by $\mathrm{s}(\mathrm{r}) = (s_1(\mathrm{r}), s_2(\mathrm{r}), ..., s_n(\mathrm{r}))$. The expected reward of reporting a probability distribution r' is $\mathrm{r} \cdot \mathrm{s}(\mathrm{r}')$, where r is the true probability distribution and r' is the reported one.

Definition 8.2.1. $\mathrm{s}(\cdot)$ *is a proper scoring rule if and only if the expected reward* $\mathrm{r} \cdot \mathrm{s}(\mathrm{r}')$ *is maximized at* $\mathrm{r}' = \mathrm{r}$.

We may additionally impose a rational participation constraint $\mathrm{r} \cdot \mathrm{s}(\mathrm{r}) \geq 0$, so that the reward for not participating is zero. It is easy to verify that the following common scoring rules are proper:

- the logarithmic scoring rule $s_i(\mathrm{r}) = a_i + b \log(r_i)$;
- the quadratic scoring rule $s_i(\mathrm{r}) = a_i + b(2r_i - \sum_k r_k^2)$;
- the spherical scoring rule $s_i(\mathrm{r}) = \frac{r_i}{\sqrt{r_1^2 + ... + r_n^2}}$.

Market scoring rules provide a way to trade proper scoring rules, in the following fashion. At any time t a public estimate r_t of the probability distribution is maintained. An agent who does not agree with the current public estimate can change it, by paying the score of that estimate and receiving the score of his new estimate r_{t+1}. The trader's expected profit from changing the probability distribution is thus $\mathrm{r} \cdot [\mathrm{s}(\mathrm{r}_{t+1}) - \mathrm{s}(\mathrm{r}_t)]$. Since the score of the previous report $\mathrm{s}(\mathrm{r}_t)$ is independent of the new report, the expected profit is maximized by maximizing the expected score of the new report, $\mathbf{r} \cdot \mathbf{s}(\mathbf{r}_{t+1})$, which is achieved by reporting the trader's true belief, $\mathbf{r}_{t+1} = \mathbf{r}$ by the definition of proper scoring rules. In other words:

Remark 8.2.2. *Truthful reporting of the probability distribution in Market Scoring Rules is myopically optimal.*

In practice, agents change the public probability vector indirectly, by buying and selling a corresponding number of shares in each security. In this equivalent definition, the score vector $\mathbf{s} = (s_1, ..., s_n)$ signifies what are the net sales in each security so far: s_i is the number of shares sold in security i, which is defined as "Pays \$1 if the outcome is i." The instantaneous price of security i can be computed as a generalized inverse of the scoring rule function. For example, the logarithmic scoring rule yields the following instantaneous price for security i:

$$c_i(\mathbf{s}) = \frac{e^{(s_i - a_i)/b}}{\sum_k e^{(s_k - a_k)/b}}. \tag{8.4}$$

To find the cost of purchasing any amount of security i, we must integrate the instantaneous price as it changes over that sale amount. Denote the total sale vector in the market as a function of time by $\mathbf{h}(t) = (h_1(t), ..., h_n(t))$, where $h_i(t)$ is the amount of security i purchased until time t. The total amount of a sale starting at time $\underline{t}$ and ending at $\bar{t}$ is $[\mathbf{h}(\bar{t}) - \mathbf{h}(\underline{t})]$ and its total cost is

$$\int_{\underline{t}}^{\bar{t}} \mathbf{c}[\mathbf{h}(t)] \cdot \mathbf{h}'(t)dt = \int_{\underline{t}}^{\bar{t}} \sum_i c_i[\mathbf{h}(t)]h_i'(t)dt.$$

Remark 8.2.3. *The integral over the instantaneous price, and consequently the cost of purchasing any amount of securities are path-independent.*

8.3 Strategic analysis of markets with a market maker

8.3.1 Description of the projection market

In this section, we describe a new abstract betting market, the projection market, as a tool for a unified and intuitive strategic analysis of prediction markets. In the following sections, we will argue that both the MSR and the DPM are strategically similar to the projection market in that a move in one market is profitable if and only if it is also profitable in the others. The projection market is conceptually simpler than MSR and DPM, and hence it is easier to analyze. We assume throughout that players are risk-neutral, namely they care about maximizing their expected profit.

Suppose there are n mutually exclusive and exhaustive events A_1,...,A_n. There are agents who may have information about the likelihood of the events A_i, and we (the designers) would like to aggregate their information. We invite them to trade in the market described below:

At any point in the trading, there is a current state described by a vector, $(x_1, x_2, ..., x_n)$, which we sometimes write in vector form as $\mathbf{x}$. Intuitively, x_i corresponds to the total holding of shares in event A_i. Each move or trade in the market can be represented by an *arrow* (or *segment*) from $\mathbf{x}$

to $\mathbf{x}'$. We use the notation $[(x_1, ..., x_n) \rightarrow (x'_1, ..., x'_n)]$ or $[\mathbf{x}, \mathbf{x}']$ to denote this move. The market starts at $(0, ..., 0)$, but the market maker makes the first move; without loss of generality, we can assume the move is to $(1, ..., 1)$. All subsequent moves are made by players, in an arbitrary (and potentially repeating) sequence.

Definition 8.3.1. *Each move has a* cost *associated with it, given by*

$$\mathcal{C}[\mathbf{x}, \mathbf{x}'] = |\mathbf{x}'| - |\mathbf{x}|,$$

where $|\cdot|$ *denotes the Euclidean norm,* $|\mathbf{x}| = \sqrt{x_1^2 + ... + x_n^2}$.

Note that none of the variables are constrained to be non-negative, and hence, the cost of a move can be negative.

The cost can be expressed in an alternative form that is also useful. Suppose player i moves from $\mathbf{x} = (x_1, ..., x_n)$ to $\mathbf{x}' = (x'_1, ..., x'_n)$. We can write $\mathbf{x}'$ as $\mathbf{x} + l\mathbf{e}$, such that $l \geq 0$ and $|\mathbf{e}| = 1$. We call l the *volume* of the move, and $\mathbf{e}$ the *direction* of the move. At any point $\hat{\mathbf{x}}$, there is an instantaneous price charged, defined as follows:

$$c(\hat{\mathbf{x}}, \mathbf{e}) = \frac{\hat{\mathbf{x}} \cdot \mathbf{e}}{|\hat{\mathbf{x}}|}.$$

Note that the price depends only on the angle between the line joining the vector $\hat{\mathbf{x}}$ and the segment $[(x_1, ..., x_n) \rightarrow (x'_1, ..., x'_n)] = [\mathbf{x}, \mathbf{x}']$, and not the lengths. The total cost of the move is the price integrated over the segment $[\mathbf{x}, \mathbf{x}']$, *i.e.*,

$$\mathcal{C}[\mathbf{x}, \mathbf{x}'] = \int_{w=0}^{l} c(\mathbf{x} + w\mathbf{e}, \mathbf{e}) \mathrm{d}w$$

We assume that trading terminates after a finite number of moves. At the end of trading, the true probabilities p_i of the event A_i are determined, and the agents receive payoffs for the moves they made. Denote the normalized probability vector $\mathbf{q} = \frac{\mathbf{p}}{|\mathbf{p}|}$.

Definition 8.3.2. *The payoff to agent i for a segment* $[\mathbf{x}, \mathbf{x}']$ *is given by:*

$$\mathcal{P}(\mathbf{x}, \mathbf{x}') = \mathbf{q} \cdot (\mathbf{x}' - \mathbf{x}).$$

Note that the payoff, too, may be negative. We call the line through the origin and point $\mathbf{p}$ the p-line. (In two dimensions, this is the line with slope $(1 - p)/p$ where p is the probability of the first event).

One drawback of the definition of a projection market is that implementing the payoffs requires us to know the actual probabilities $p_1, ..., p_n$. This is feasible if the probability can eventually be determined statistically, such as when predicting the relative frequency of different recurring events, or vote shares. It is also feasible for one-off events in which there is reason to believe that the true probability is either 0 or 1. For other one-off events, it cannot be implemented directly (unlike scoring rules, which can be implemented in expectation). However, we believe that even in these cases, the projection market can be useful as a conceptual and analytical tool.

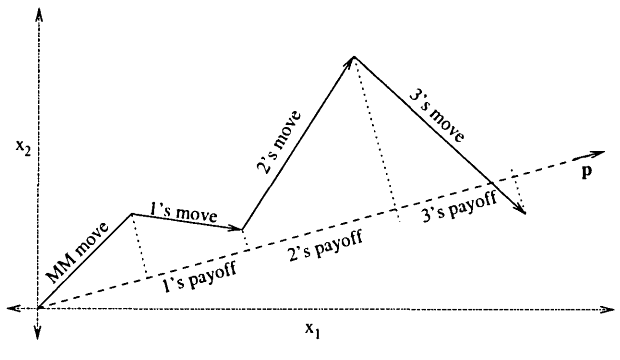


Figure 8.1. Trade in a projection market with three traders and two events.

The moves, costs and payoffs have a natural geometric representation, which is shown in Figure 8.1 for three players with one move each in a world with two events. The players append directed line segments in turn, and the payoff player i finally receives for a move is the projection of her segment onto the line with slope $(1 - p)/p$ (or generally, the p-line). Her cost is the difference of distances of the endpoints of her move to the origin.

■ 8.3.2 Strategic properties of the projection market

We begin our strategic analysis of the projection market by observing the following simple path-independence property.

Lemma 8.3.3. [Path-Independence] *Suppose there is a sequence of moves leading from* x *to* x′*. Then, the total cost of all the moves is equal to the cost of the single move* $[\mathrm{x} \rightarrow \mathrm{x}']$*, and the total payoff of all the moves is equal to the payoff of the single move* $[\mathrm{x} \rightarrow \mathrm{x}']$.

Proof. The proof follows trivially from the definition of the costs and payoffs: If we consider a path from point x to point x′, both the net change in the vector lengths and the net projection onto the p-line are completely determined by x and x′. □

Although simple, path independence of profits is vitally important, because it implies (and is implied by) the absence of arbitrage in the market. In other words, there is no sequence of moves that start and end at the same point, but result in a positive profit. On the other hand, if there were two paths from x to x′ with different profits, there would be a cyclic path with positive profit.

For ease of reference, we summarize some more useful properties of the cost and payoff functions in the projection market.

Lemma 8.3.4.

1. *The instantaneous price for moving along a line through the origin is* 1 *or* -1*, when the move is away or toward the origin respectively. The instantaneous price along a circle centered at the origin is* 0.

2. *When* x *moves along a circle centered at the origin to point* $\bar{x}$ *on the positive* p*-line, the corresponding payoff is* $\mathcal{P}(x, \bar{x}) = |x| - x \cdot q$*, and the cost is* $\mathcal{C}[x, \bar{x}] = 0$.

3. *The two cost function formulations are equivalent:*

$$\mathcal{C}[x, x'] = \int_{w=0}^{l} \cos(x + we, e)dw = |x'| - |x| \qquad \forall x, x',$$

where e *is the unit vector giving the direction of move. In addition, when* x *moves along the positive* p*-line, the payoff is equal to the cost,* $\mathcal{P}(x, x') = |x'| - |x|$.

x',

Proof. 1. The instantaneous price is

$$c(x, e) = x \cdot e/|x| = \cos(x, e),$$

where e is the direction of movement, and the result follows.

2. Since $\bar{x}$ is on the positive p-line, $q \cdot \bar{x} = |\bar{x}| = |x|$, hence $\mathcal{P}(x, \bar{x}) = q \cdot (\bar{x} - x) = |x| - x \cdot q$; the cost is 0 from the definition.

3. From Part 1, the cost of moving from x to the origin is

$$\mathcal{C}[x, 0] = \int_{w=0}^{l} \cos(x + we, e)\mathrm{dw} = \int_{w=0}^{l} (-1)\mathrm{dw} = -|x|,$$

where $l = |x|$, $e = x/|x|$. By the path-independence property,

$$\mathcal{C}[x, x'] = \mathcal{C}[x, 0] + \mathcal{C}[0, x'] = |x'| - |x|.$$

Finally, a point on the positive p-line gets projected to itself, namely $q \cdot x = |x|$ so when the movement is along the positive p-line, $\mathcal{P}(x, x') = q \cdot (x' - x) = |x'| - |x| = \mathcal{C}[x, x']$.

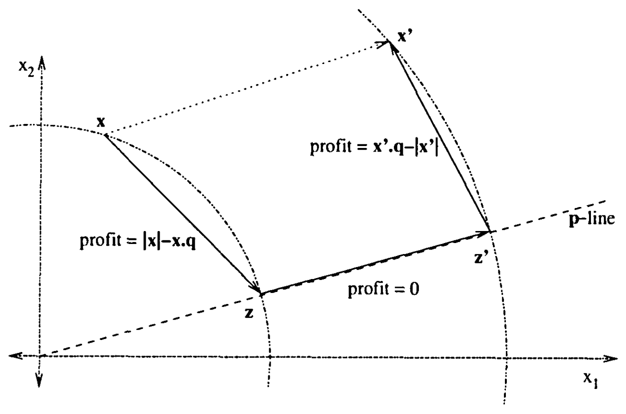
□

We now consider the question of which moves are profitable in this market. The eventual profit of a move $[x, x']$, where $x' = x + le$, is

$$\begin{aligned} profit[x, x'] &= \mathcal{P}[x, x'] - \mathcal{C}[x, x'] \\ &= lq.e - \mathcal{C}[x, x'] \end{aligned}$$

Differentiating with respect to l, we get

$$\begin{aligned} \frac{d(profit)}{dl} &= q.e - c(x + le, e) \\ &= q.e - \frac{x + le}{|x + le|}.e \end{aligned}$$



Proof.

Figure 8.2. The profit of move $[\mathrm{x}, \mathrm{x}']$ is equal to the change in profit potential from x to x'.

We observe that this is 0 if the vectors q and $(\mathrm{x} + l\mathrm{e})$ are exactly aligned. Further, we observe that the price is non-decreasing with increasing l. Thus, along the direction e, the profit is maximized at the point of intersection with the p-line.

By Lemma 8.3.4, there is always a path from x to the positive p-line with 0 cost, which is given by an arc of the circle with center at the origin and radius $|\mathrm{x}|$. Also, any movement along the p-line has 0 additional profit. Thus, for any point x, we can define the profit potential $\phi(\mathrm{x}, p)$ by

$$\phi(\mathrm{x}, p) = |\mathrm{x}| - \mathrm{x} \cdot \mathrm{q}.$$

Note, the potential is positive for x off the positive p-line and zero for x on the line. Next we show that a move to a lower potential is always profitable.

Lemma 8.3.5. *The profit of a move* $[\mathrm{x}, \mathrm{x}']$ *is equal to the difference in potential* $\phi(\mathrm{x}, p) - \phi(\mathrm{x}', p)$.

Denote $\mathrm{z} = |\mathrm{x}|\mathrm{q}$ and $\mathrm{z}' = |\mathrm{x}'|\mathrm{q}$, *i.e.,* these are the points of intersection of the positive p-line with the circles centered at the origin with radii $|\mathrm{x}|$ and $|\mathrm{x}'|$ respectively. By the path-independence property and Lemma 8.3.4, the profit of move $[\mathrm{x}, \mathrm{x}']$ is

$$\begin{aligned} profit(\mathrm{x}, \mathrm{x}') &= profit(\mathrm{x}, \mathrm{z}) + profit(\mathrm{z}, \mathrm{z}') + profit(\mathrm{z}', \mathrm{x}') \\ &= (|\mathrm{x}| - \mathrm{x} \cdot \mathrm{q}) + 0 + (\mathrm{x}' \cdot \mathrm{q} - |\mathrm{x}'|) \\ &= \phi(\mathrm{x}, p) - \phi(\mathrm{x}', p). \end{aligned}$$

Thus, the profit of the move is equal to the change in profit potential between the endpoints. □

This lemma offers another way of seeing that it is optimal to move to the point of lowest potential, namely to the p-line.

8.4 Strategic equivalence of projection market to dynamic parimutuel markets

In this section, we show that the dynamic parimutuel market is remarkably similar to the projection market: we demonstrate that there is a one-to-one correspondence between moves in the two markets, that have equal cost, payoff and profit. Coupled with section 8.5, this strategic equivalence also demonstrates a strong connection between the DPM and MSR.

Cost of a trade in the DPM Consider a trader who comes to a DPM in state $(M, \mathbf{x})$, and buys or sells shares such that the eventual state is $(M', \mathbf{x}')$. What is the net cost, $M' - M$, of her move?

Recall from Section 8.1 that in the share-ratio DPM, the instantaneous prices are defined by the principle that the ratio of prices of two securities equals the ratio of total number of shares purchased in each security. In particular, this yields the following formula for the instantaneous costs:

$$c_i = x_i \frac{M}{|\mathbf{x}|^2} \qquad \forall i,$$

whereby we concluded in Theorem 8.1.1 that the net cost of a move $[\mathbf{x}, \mathbf{x}']$ is

$$M' - M = M_0[|\mathbf{x}'| - |\mathbf{x}|],$$

for some constant M_0. In other words, the cost of a move is a constant multiple of the corresponding cost in the projection market.

Constraints in the DPM Although it might seem, based on the costs, that any move in the projection market has an equivalent move in the DPM, the DPM places some constraints on trades. Firstly, no trader is allowed to have a net negative holding in either share. This is important, because it ensures that the total holdings in each share are always positive. However, this is a boundary constraint, and does not impact the strategic choices for a player with a sufficiently large positive holding in each share. Thus, we can ignore this constraint from a first-order strategic analysis of the DPM. Secondly, for practical reasons a DPM will probably have a minimum unit of trade, but we assume here that arbitrarily small quantities can be traded.

Payoffs in the DPM At some point, trading in the DPM ceases and shares are liquidated. We assume here that the true probability becomes known at liquidation time, and describe the payoffs in terms of the probability; however, if the probability is not revealed, only the event that actually occurs, these payoffs can be implemented in expectation. Suppose the DPM terminates in a state $(M, \mathbf{x})$, and the true probability of event A_i is p_i. When the dynamic parimutuel market is liquidated, the shares are paid off in the following way: Each owner of a share of A_i receives $p_i \frac{M}{x_i}$ for each share owned.

The payoffs in the DPM, although given by a fairly simple form, are conceptually complex, because the payoff of a move depends on the subsequent moves before the market liquidates. Thus, a fully rational choice of move in the DPM for a player should take into account the actions of subsequent players, including that player herself.

Here, we restrict the analysis to myopic, infinitesimal strategies: Given the market position is $(M, \mathbf{x})$, in which direction should a player make an infinitesimal move in order to maximize her profit? We show that the infinitesimal payoffs and profits of a DPM with true probability $\mathbf{p}$ correspond strategically to the infinitesimal payoffs and profits of a projection market with normalized probability vector $(\sqrt{p_1}, \ldots, \sqrt{p_n})$, in the following sense:

Lemma 8.4.1. *Suppose a player is about to make a move in a dynamic parimutuel market in a state $(M, \mathbf{x})$, and the true probability of event A_i is p_i. Then, assuming the market is liquidated after that player's move,*

- *If $\frac{x_i}{|\mathbf{x}|} < \sqrt{p_i}$, player i profits by buying shares in A_i.*
- *If $\frac{x_i}{|\mathbf{x}|} > \sqrt{p_i}$, player i profits by selling shares in A_i.*

Proof. Consider the cost and payoff of buying a small quantity Δx_i of shares in A_i. The cost is $C[(x_1, ..., x_i, ..., x_n \rightarrow (x_1, ..., x_i + \Delta x_i, ..., x_n)] = \Delta x_i \cdot x_i \frac{M}{|\mathbf{x}|^2}$, and the payoff is $\Delta x_i \cdot p_i \frac{M}{x_i}$. Thus, buying the shares is profitable iff

$$\begin{aligned}
\Delta x_i \cdot x_i \frac{M}{|\mathbf{x}^2|} &< \Delta x_i \cdot p_i \frac{M}{x_i} \\
\Leftrightarrow \frac{x_i^2}{|\mathbf{x}|^2} &< p_i \\
\Leftrightarrow \frac{x_i}{|\mathbf{x}|} &< \sqrt{p_i}
\end{aligned}$$

Thus, buying A is profitable if $\frac{x_i}{\mathbf{x}} < \sqrt{p_i}$, and selling A is profitable if $\frac{x_i}{\mathbf{x}} > \sqrt{p_i}$. □

It follows from Lemma 8.4.1 that it is myopically profitable for players to move towards the line with slope $(\sqrt{p_1}, ..., \sqrt{p_n})$. Note that there is a one-to-one mapping between $(p_1, ..., p_n)$ and $(\sqrt{p_1}, ..., \sqrt{p_n})$ in their respective ranges, so this line is uniquely defined: each such line also corresponds to a unique vector $\mathbf{p}$. However, because the actual payoff of a move depends on the future moves, players must base their decisions on some belief about the final state of the market. In the light of Lemma 8.4.1, one natural, rational-expectation style assumption is that the final state $(M, \mathbf{x}^*)$ will satisfy $\frac{x_i^*}{|\mathbf{x}^*|} = \sqrt{p_i}$. (In other words, one might assume that the traders' beliefs will ultimately converge to the true probability p_i; knowing p_i, the traders will drive the market state to satisfy $\frac{x}{|\mathbf{x}|} = \sqrt{p_i}$.) This is very plausible in markets (such as the Yahoo! Buzz market) in which trading is permitted right until the market is liquidated, at which point there is no remaining uncertainty about the relevant frequencies. Under this assumption, we can prove an even tighter connection between payoffs in the DPM (where the true probability is $\mathbf{p}$) and payoffs in the projection market, with odds $(\sqrt{p_1}, ..., \sqrt{p_n})$:

Theorem 8.4.2. *Suppose that the DPM ultimately terminates in a state $(M, \mathbf{X})$ satisfying $\frac{X_i}{|\mathbf{X}|} = \sqrt{p_i}$. Assume without loss of generality that the constant $M_0 = 1$, so $M = |\mathbf{X}|$. Then, the final payoff for any move $[\mathbf{x} \rightarrow \mathbf{x}']$ made in the course of trading is $(\mathbf{x}' - \mathbf{x}) \cdot (\sqrt{p_1}, ..., \sqrt{p_n})$, i.e., it is the same as the payoff in the projection market with odds $(\sqrt{p_1}, ..., \sqrt{p_n})$.*

Proof. First, observe that $\frac{X_i}{M} = \sqrt{p_i}$. The final payoff is the liquidation value of $(x_i' - x_i)$ shares of A_i for all i, which is

$$\begin{aligned}
\text{Payoff}_{DPM}[\mathbf{x}' - \mathbf{x}] &= \sum_{i=1}^{n} p_i \frac{M}{X_i}(x_i' - x_i) \\
&= \sum_{i=1}^{n} p_i \frac{1}{\sqrt{p_i}}(x_i' - x_i) \\
&= \sum_{i=1}^{n} \sqrt{p_i}(x_i' - x_i).
\end{aligned}$$

□

Strategic Analysis for the DPM Theorems 8.1.1 and 8.4.2 give us a very strong equivalence between the projection market and the dynamic parimutuel market, under the assumption that the DPM converges to the optimal value for the true probability. A player playing in a DPM with true odds $(p_1, ..., p_n)$, can imagine himself playing in the projection market with odds $(\sqrt{p_1}, ..., \sqrt{p_n})$, because both the costs and the payoffs of any given move are identical.

Using this equivalence, we can transfer all the strategic properties proven for the projection market directly to the analysis of the dynamic parimutuel market. One particularly interesting conclusion we can draw is as follows: In the absence of any constraint that disallows it, it is always profitable for an agent to move towards the origin, by selling shares in all events A_i while maintaining the ratio $x_i / \sum_j x_j$. In the DPM, this is limited by forbidding short sales, so players can never have negative holdings in any share. As a result, when their holding in one share (say A_i) is 0, they can't use the strategy of moving towards the origin. We can conclude that a rational player should never hold shares of *all* A_i simultaneously, regardless of her beliefs and the market position.

This discussion leads us to consider a modified DPM, in which this strategic loophole is addressed directly: Instead of disallowing all short sales, we place a constraint that no agent ever reduce the total market capitalization M (or, alternatively, that any agent's total investment in the market is always non-negative). We call this the "nondecreasing market capitalization" constraint for the DPM. This corresponds to a restriction that no move in the projection market reduces the radius. However, we can conclude from the preceding discussion that players have no incentive to ever increase the radius. Thus, the moves of the projection market would all lie on the quarter sphere in the positive quadrant, with radius determined by the market maker's move. In section 8.5, we show that the projection market on this quarter sphere is strategically equivalent (at least myopically) to trade in a Market Scoring Rule. Thus, the DPM and MSR appear to be deeply connected to each other, like different interfaces to the same underlying market.

8.5 The projection market vs market scoring rules

8.5.1 Strategic equivalence to the spherical market scoring rule

In this section, we demonstrate a close connection between the projection market restricted to a circular arc and a market scoring rule that uses the spherical scoring rule.

Definition 8.5.1. *The* spherical scoring rule *[46] is defined by*

$$s_i(\mathbf{r}) = \frac{r_i}{|\mathbf{r}|} = \frac{r_i}{\sqrt{r_1^2 + ... + r_n^2}}.$$

The spherical scoring rule is known to be a proper scoring rule. At this point, it is convenient to use vector notation. Let $\mathrm{x} = (x_1, ..., x_n)$ denote a position in the projection market. We consider the projection market restricted to the circle $|\mathrm{x}| = 1$.

Restricted projection market Consider a move in this restricted projection market from x to x'. Recall that $\mathrm{q} = \frac{\mathrm{p}}{|\mathrm{p}|}$, where $\mathrm{p} = (p_1, ..., p_n)$ is the true probability vector of events $A_1, ..., A_n$. Then, the projection market profit of a move $[\mathrm{x}, \mathrm{x}']$ is $\mathrm{q} \cdot [\mathrm{x}' - \mathrm{x}]$ (noting that $|\mathrm{x}| = |\mathrm{x}'|$).

We can extend this to an arbitrary collection[1] of (not necessarily contiguous) moves $\mathcal{X} = \{[\mathrm{x}_1, \mathrm{x}_1'], [\mathrm{x}_2, \mathrm{x}_2'], \cdots, [\mathrm{x}_l, \mathrm{x}_l']\}$.

$$\begin{aligned}\text{PROJ-PROFIT}_p(\mathcal{X}) &= \sum_{[\mathrm{x},\mathrm{x}']\in\mathcal{X}} \mathrm{q} \cdot [\mathrm{x}' - \mathrm{x}] \\ &= \mathrm{q} \cdot \left[\sum_{[\mathrm{x},\mathrm{x}']\in\mathcal{X}} [\mathrm{x}' - \mathrm{x}] \right].\end{aligned}$$

Spherical scoring rule profit We now turn our attention to the MSR with the spherical scoring rule (SSR). Consider a player who changes the report from r to r'. Then, if the true probability of A_i is p_i, her expected profit is

$$\text{SSR-PROFIT}([\mathrm{r}, \mathrm{r}']) = \sum_{i=1}^{n} p_i(s_i(\mathrm{r}') - s_i(\mathrm{r})).$$

Now, let us represent the initial and final position in terms of spherical coordinates. For $\mathrm{r} = (r_1, ..., r_n)$, define the corresponding coordinates $\mathrm{x} = \frac{\mathrm{r}}{|\mathrm{r}|} = (\frac{r_1}{|\mathrm{r}|}, ..., \frac{r_n}{|\mathrm{r}|})$. Note that the coordinates satisfy $|\mathrm{x}| = 1$, and thus correspond to valid coordinates for the restricted projection market.

Now, let p denote the vector $(p_1, ..., p_n)$. Then, expanding the spherical scoring functions $s_i(.)$, the player's profit for a move from r to r' can be rewritten in terms of the corresponding coordinates x, x' as:

$$\text{SSR-PROFIT}([\mathrm{x}, \mathrm{x}']) = \mathrm{p} \cdot (\mathrm{x}' - \mathrm{x}).$$

For any collection $\mathcal{X}$ of moves, the total payoff in the SSR market is given by:

$$\begin{aligned}\text{SSR-PROFIT}_p(\mathcal{X}) &= \sum_{[\mathrm{x},\mathrm{x}']\in\mathcal{X}} \mathrm{p} \cdot [\mathrm{x}' - \mathrm{x}] \\ &= \mathrm{p} \cdot \left[\sum_{[\mathrm{x},\mathrm{x}']\in\mathcal{X}} [\mathrm{x}' - \mathrm{x}] \right].\end{aligned}$$

[1]We allow the collection to contain repeated moves, *i.e.*, it is a multiset.

Finally, we note that $\mathbf{p}$ and $\mathbf{q}$ are related by $\mathbf{q} = \mu_p \mathbf{p}$, where $\mu_p = \frac{1}{|\mathbf{p}|}$ is a scalar that depends only on $\mathbf{p}$. This immediately gives us the following strong strategic equivalence for the restricted projection market and the SSR market:

Theorem 8.5.2. *Any collection of moves $\mathcal{X}$ yields a positive (negative) payoff in the restricted projection market iff $\mathcal{X}$ yields a positive (negative) payoff in the Spherical Scoring Rule market.*

Proof. As derived above,

$$\text{PROJ-PROFIT}_p(\mathcal{X}) = \mu_p \text{SSR-PROFIT}_p(\mathcal{X}).$$

For all $\mathbf{p}$, $1 \leq \mu_p = \frac{1}{|\mathbf{p}|} \leq \sqrt{n}$, by the arithmetic mean—root mean square inequality, and the result follows immediately. □

Although theorem 8.5.2 is stated in terms of the sign of the payoff, it extends to relative payoffs of two collections of moves:

Corollary 8.5.3. *Consider any two collections of moves $\mathcal{X}$, $\mathcal{X}'$. Then, $\mathcal{X}$ yields a greater payoff than $\mathcal{X}'$ in the projection market iff $\mathcal{X}$ yields a greater payment than $\mathcal{X}'$ in the SSR market.*

Proof. Every move $[\mathbf{x}, \mathbf{x}']$ has a corresponding inverse move $[\mathbf{x}', \mathbf{x}]$. In both the projection market and the SSR, the inverse move profit is simply the negative profit of the move (the moves are reversible). We can define a collection of moves $\mathcal{X}'' = \mathcal{X} - \mathcal{X}'$ by adding the inverse of $\mathcal{X}'$ to $\mathcal{X}$. Note that

$$\text{PROJ-PROFIT}_p(\mathcal{X}'') = \text{PROJ-PROFIT}_p(\mathcal{X}) - \text{PROJ-PROFIT}_p(\mathcal{X}')$$

and

$$\text{SSR-PROFIT}_p(\mathcal{X}'') = \text{SSR-PROFIT}_p(\mathcal{X}) - \text{SSR-PROFIT}_p(\mathcal{X}');$$

applying theorem 8.5.2 completes the proof. □

It follows that the *ex post* optimality of a move (or set of moves) is the same in both the projection market and the SSR market. On its own, this strong *ex post* equivalence is not completely satisfying, because in any non-trivial market there is uncertainty about the value of $\mathbf{p}$, and the different scaling ratios for different $\mathbf{p}$ could lead to different *ex ante* optimal behavior. We can extend the correspondence to settings with uncertain $\mathbf{p}$, as follows. Denote the probability simplex $\Delta = \{\mathbf{y} \mid 0 \leq y_i \leq 1;\ \sum_{i=1}^{n} y_i = 1\}$.

Theorem 8.5.4. *Consider the restricted projection market with some prior probability distribution F over possible values of* $\mathbf{p}$. *Then, there is a probability distribution G with the same support as F, and a strictly positive constant c that depends only on F such that:*

- *(i) For any collection $\mathcal{X}$ of moves, the expected profits are related by:*

$$E_F(\textit{PROJ-PROFIT}(\mathcal{X})) = cE_G(\textit{SSR-PROFIT}(\mathcal{X}))$$

- *(ii) For any measurable information set $I \subseteq \Delta$, the expected profits of any collection $\mathcal{X}$ of moves, conditioned on knowing that* $\mathbf{p} \in I$, *satisfy*

$$E_F(\textit{PROJ-PROFIT}(\mathcal{X})|\mathbf{p} \in I) = c_I E_G(\textit{SSR-PROFIT}(\mathcal{X})|\mathbf{p} \in I)$$

The converse also holds: For any probability distribution G, there is a distribution F such that both these statements are true.

Proof. For simplicity, assume that F has a density function f. (The result holds even for non-continuous distributions). Then, let $c = \int_{\mathbf{p}\in\Delta} \mu_p f(\mathbf{p}) d\mathbf{p}$. Define the density function g of distribution G by

$$g(\mathbf{p}) = \frac{\mu_p f(\mathbf{p})}{c}$$

Now, for a collection of moves $\mathcal{X}$,

$$\begin{aligned}
E_F(\text{PROJ-PROFIT}(\mathcal{X})) &= \int \text{PROJ-PROFIT}_p(\mathcal{X}) f(\mathbf{p}) d\mathbf{p} \\
&= \int \text{SSR-PROFIT}_p(\mathcal{X}) \mu_p f(\mathbf{p}) d\mathbf{p} \\
&= \int \text{SSR-PROFIT}_p(\mathcal{X}) c g(\mathbf{p}) d\mathbf{p} \\
&= c E_G(\text{SSR-PROFIT}(\mathcal{X}))
\end{aligned}$$

To prove part (ii), we simply restrict the integral to values in I, and normalize by the probability of I; the normalization may be slightly different under distributions F and G, which is why the constant c_I depends on I. The important point is that it is independent of $\mathcal{X}$. The converse follows similarly by constructing F from G. □

Analysis of MSR strategies Theorem 8.5.4 provides the foundation for analysis of strategies in scoring rule markets. To the extent that strategies in these markets are independent of the specific scoring rule used, we can use the spherical scoring rule as the market instrument. Then, analysis of strategies in the projection market with a slightly distorted distribution over $\mathbf{p}$ can be used to understand the strategic properties of the original market situation.

Implementation in expectation Another important consequence of Theorem 8.5.4 is that the *restricted* projection market can be implemented with a small distortion in the probability distribution over values of $\mathbf{p}$, by using a Spherical Scoring Rule to implement the payoffs. This makes the projection market valuable as a design tool; for example, we can analyze new constraints and rules in the projection market, and then implement them via the SSR.

■ 8.5.2 Relaxing the restrictions

Section 8.5.1 contained a strong equivalence result for the projection market restricted to a sphere and the spherical market scoring rule. In this section, we describe a scheme to decompose moves in the unrestricted projection market into two simpler sets of moves, and use this to derive weaker restrictions under which the equivalence holds.

Consider a move $[\mathbf{z}, \mathbf{z}']$ in the unrestricted projection market. Given any fixed unit vector $\mathbf{y}$, this move can be decomposed into three moves $[\mathbf{z}, |\mathbf{z}|\mathbf{y}]$, $[|\mathbf{z}|\mathbf{y}, |\mathbf{z}'|\mathbf{y}]$, and $[|\mathbf{z}'|\mathbf{y}, \mathbf{z}']$ by path independence. It follows that, for any collection $\mathcal{Z} = \{[\mathbf{z}_i, \mathbf{z}'_i\}$, we can decompose $\mathcal{Z}$ into the union of two collections $\mathcal{X}, \mathcal{Y}$ such that:

$$\begin{aligned} \forall[\mathbf{x}_i, \mathbf{x}'_i] \in \mathcal{X} \quad &, |\mathbf{x}_i| = |\mathbf{x}'_i| = \alpha_i \\ \forall[\mathbf{y}_j, \mathbf{y}'_j] \in \mathcal{Y} \quad &, \mathbf{y}_j = \beta_j \mathbf{y} \text{ and } \mathbf{y}'_j = \beta'_j \mathbf{y} \end{aligned}$$

Consider a pair of vectors $(\mathbf{x}_i, \mathbf{x}'_i)$ with $|\mathbf{x}_i| = |\mathbf{x}'_i| = \alpha_i$, and let $\mathbf{p}$ be any probability vector. Then, $\mathbf{p} \cdot (\mathbf{x}'_i - \mathbf{x}_i) = \alpha_i \mathbf{p} \cdot (\frac{\mathbf{x}'_i}{\alpha_i}, \frac{\mathbf{x}_i}{\alpha_i})$ corresponds to a weighted market spherical scoring rule, *i.e.*, , a simple scaling of the market SSR by α_i. Thus, we can extend the notation SSR-PROFIT$(\mathcal{X})$ to collections of moves in which each move is on a spherical arc (not necessarily of radius 1), to denote the sum of the correspondingly weighted SSR profits.

Now, suppose we are given fixed parameter vectors $\boldsymbol{\beta} = [\beta_j], \boldsymbol{\beta}' = [\beta'_j]$.

Definition 8.5.5. *A collection of moves $\mathcal{Z}$ is* $(\boldsymbol{\beta}, \boldsymbol{\beta}')$-consistent *if it can be decomposed into collections $\mathcal{X}$ and $\mathcal{Y}$ as described above such that the moves in $\mathcal{Y}$ have the corresponding values of β_j and β'_j.*

Observe that the $(\boldsymbol{\beta}, \boldsymbol{\beta}')$-consistency of $\mathcal{Z}$ is independent of the choice of unit vector $\mathbf{y}$.

We can now state and prove a generalization of theorem 8.5.4.

Theorem 8.5.6. *Suppose fixed $\boldsymbol{\beta}, \boldsymbol{\beta}'$ and any unit vector* $\mathbf{y}$ *are given. Consider the restricted projection market with some prior probability distribution F over possible values of* $\mathbf{p}$. *Then, there is a probability distribution G with the same support as F, and a strictly positive constant c that depends only on F such that:*
For any $(\boldsymbol{\beta}, \boldsymbol{\beta}')$-consistent collection $\mathcal{Z}$ of moves, the expected profits are related by:

$$E_F(\textit{PROJ-PROFIT}(\mathcal{Z})) = cE_G(\textit{SSR-PROFIT}(\mathcal{X})) + d$$

where c, d that are independent of $\mathcal{Z}$.

Proof. Note that PROJ-PROFIT$(\mathcal{Z})$ = PROJ-PROFIT$(\mathcal{X})$ + PROJ-PROFIT$(\mathcal{Y})$. For G defined as in the proof of theorem 8.5.4, we have E_F(PROJ-PROFIT$(\mathcal{X})$) = cE_G(PROJ-PROFIT$(\mathcal{X})$), where c depends only on F. Further,

$$E_F(\text{PROJ-PROFIT}(\mathcal{Y})) = E_F(\mathbf{q} \cdot \mathbf{y} \sum_j (\beta_j - \beta'_j))$$

is independent of $\mathcal{Z}$ as long as $\boldsymbol{\beta}, \boldsymbol{\beta}'$ are fixed. □

Note that the restriction to a sphere is an extreme special case of a restriction to $(\boldsymbol{\beta}, \boldsymbol{\beta}')$-consistent collections of moves, with $\boldsymbol{\beta}$ and $\boldsymbol{\beta}'$ set to null vectors. As stated, theorem 8.5.6 generalizes only one of the results of theorem 8.5.4, but the converse and the restriction to a given information set follow as easily.

Implications Theorem 8.5.6 shows that the strategic equivalence (with a slightly distorted prior) between the projection market and the market SSR extends to situations in which an agent is constrained to choose between (β, β')-consistent collections of moves $\mathcal{Z}$. We can achieve this by ensuring that the radius of the final position of each move the agent makes is predetermined, *even if it is not exactly* 1. In other words, the radius (or profit scaling factor) can be varied as long as it is not under the strategic control of the agents. Theorem 8.5.6 then gives support for using the projection market to design and analyze such restrictions, which may be implemented using scaled SSRs. In section 8.6, we consider two ways in which varying the radius in this way might be useful in practice.

8.5.3 Strategic connections to other market scoring rules

We have shown a strong strategic equivalence between the projection market and the MSR based on the spherical scoring rule. In this section, we briefly describe a weaker similarity between the projection market restricted to a sphere and other market scoring rules.

The similarity is based on the concept of straight-line monotonicity of scoring rules [96]. Consider a true probability vector $\mathbf{p}$ and a reported probability vector $\mathbf{r}$. For any scoring rule $\mathrm{s}(\mathbf{r})$, let $S_{\mathbf{p}}(\mathbf{r}) = \sum_i p_i s_i(\mathbf{r})$ denote the expected score of a report $\mathbf{r}$ when the true distribution is $\mathbf{p}$. Nau proved that all proper scoring rules have the following property:

$$\forall \mathbf{p}, \mathbf{r} \text{ s.t. } \mathbf{p} \neq \mathbf{r}, \forall \lambda \in (0,1),\ S_{\mathbf{p}}(\lambda \mathbf{p} + (1-\lambda)\mathbf{r}) > S_{\mathbf{p}}(\mathbf{r})$$

This can be interpreted in the market scoring rule context as saying that for any proper scoring rule and any position $\mathbf{r}$, it is always profitable to move along the straight line towards the true probability $\mathbf{p}$.

This is easily seen to also be true for the projection market on a sphere. First, consider moving along the *chord* from any position $\mathbf{x}$ towards the true normalized probability vector $\mathbf{q}$. We have:

$$\text{PROJ-PROFIT}([\mathbf{x}, \lambda \mathbf{q} + (1-\lambda)\mathbf{x}]) = \lambda[\mathbf{q} \cdot \mathbf{q} - \mathbf{q} \cdot \mathbf{x}] > 0$$

For the projection market restricted to a sphere, moving along the chord is not legal; instead, one would have to move along the *arc* from $\mathbf{x}$ to $\mathbf{q}$. However, the projection of a position on the chord onto $\mathbf{q}$ is dominated by the projection of the corresponding position on the arc, and so the latter move is also profitable.

This is a weak similarity result, because most moves in the markets of interest will not be directly towards the (as yet unknown) true probability vector. It would be nicer if we could prove that any move that gets closer to the true probability is profitable in both the projection market and in all MSRs. However, there is no metric definition of "closer" for which this is uniformly true for all proper scoring rules [46, 96], and hence such a result is not possible in general. One special case is in the two-dimensional setting, in which all moves are on the straight line towards or away from the true probability.

8.6 Using the projection-market model for analyzing non-myopic and noisy trade

The chief advantages of the projection market are that it is analytically tractable, and also easy to visualize. In Section 8.4, we used the projection-market model of the DPM to prove the absence of arbitrage, and to infer strategic properties that might have been difficult to deduce otherwise. In this section, we provide two examples that illustrate the power of projection-market analysis for gaining insight about more complex strategic settings.

8.6.1 Traders with inertia

The standard analysis of the trader behavior in any of the market forms we have studied asserts that traders who disagree with the market probabilities will expect to gain from changing the probability, and thus have a strict incentive to trade in the market. However, the expected gain may be very small. A plausible model of real trader behavior might include some form of inertia or ϵ-optimality: We assume that traders will trade if their expected profit is greater than some constant ϵ. We do not attempt to justify this model here; rather, we illustrate how the projection market may be used to analyze such situations, and shed some light on how to modify the trading rules to alleviate this problem.

Consider the simple projection market restricted to a circular arc with unit radius; as we have seen, this corresponds closely to the spherical market scoring rule, and to the dynamic parimutuel market under a reasonable constraint. Now, suppose the market probability is p, and a trader believes the true probability is p'. Then, his expected gain can be calculated, as follows: Let q and q$'$ be the unit vectors in the directions of p and p' respectively. The expected profit is given by $E = \phi(\mathrm{q}, p') = 1 - \mathrm{q} \cdot \mathrm{q}'$. Thus, the trader will trade only if $1 - \mathrm{q} \cdot \mathrm{q}' > \epsilon$. If we let θ and θ' be the angles of the p-line and p'-line respectively (from the x-axis), we get $E = 1 - cos(\theta - \theta')$; when θ is close to θ', a Taylor series approximation gives us that $E \approx (\theta - \theta')^2/2$. Thus, we can derive a bound on the limit of the market accuracy: The market price will not change as long as $(\theta - \theta')^2 \leq 2\epsilon$.

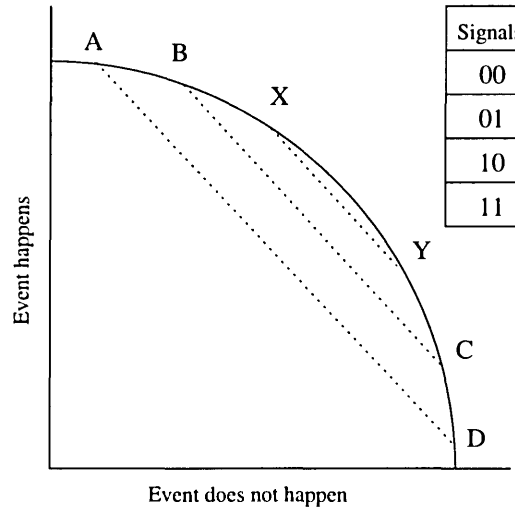
Now, suppose a market operator faced with this situation wanted to sharpen the accuracy of the market. One natural approach is simply to multiply all payoffs by a constant. This corresponds to using a larger circle in the projection market, and would indeed improve the accuracy. However, it will also increase the market-maker's exposure to loss: the market-maker would have to pump in more money to achieve this.

The projection market model suggests a natural approach to improving the accuracy while retaining the same bounds on the market maker's loss. The idea is that, instead of restricting all moves to being on the unit circle, we force each move to have a slightly *larger* radius than the previous move. Suppose we insist that, if the current radius is r, the next trader has to move to $r + 1$. Then, the trader's expected profit would be $E = r(1 - cos(\theta - \theta'))$. Using the same approximation as above, the trader would trade as long as $(\theta - \theta')^2 > 2\epsilon/r$. Now, even if the market maker seeded the market with $r = 1$, it would increase with each trade, and the incentives to sharpen the estimate increase with every trade.

8.6.2 Analyzing long-range strategies

Up to this point, our analysis has been restricted to trader strategies that are myopic in the sense that traders do not consider the impact of their trades on other traders' beliefs. In practice, an informed trader can potentially profit by playing a suboptimal strategy to mislead other traders, in a way that allows her to profit later. In this section, we illustrate how the projection market can be used to analyze an instance of this phenomenon, and to design market rules that mitigate this effect.

The scenario we consider is the following: There are two traders speculating on the probability of an event E, who each get a 1-bit signal. The optimal probability for each 2-bit signal pair is defined as follows: If trader 1 gets the signal 0, and trader 2 gets signal 0, the optimal probability is 0.3. If trader 1 got a 0, but trader 2 got a 1, the optimal probability is 0.9. If trader 1 gets 1, and trader 2 gets signal 0, the optimal probability is 0.7. If trader 1 got a 0, but trader 2 got a 1, the optimal probability is 0.1. (Note that the impact of trader 2's signal is in a different direction, depending on trader 1's signal). Suppose that the prior distribution of the signals is that trader 1 is equally likely to get a 0 or a 1, but trader 2 gets a 0 with probability 0.55 and a 1 with probability 0.45. The traders are playing the projection market restricted to a circular arc. This setup is depicted in Figure 8.3.



Signals	Opt.pt.
00	C
01	A
10	B
11	D

Figure 8.3. Example illustrating non-myopic deception

Suppose that, for some exogenous reason, trader 1 has the opportunity to trade, followed by trader 2. Then, trader 1 has the option of placing a last-minute trade just before the market closes. If traders were playing their myopically optimal strategies, here is how the market should run: If trader 1 sees a 0, he would move to some point Y that is between A and C, but closer to C. Trader 2 would then infer that trader 1 received a 0 signal and move to A or C if she got 1 or 0 respectively. Trader 1 has no reason to move again. If trader 1 had got a 1, he would move to a different point X instead, and trader 2 would move to D if she saw 1 and B if she saw 0. Again, trader 1 would not want to move again.

Using the projection market, it is easy to show that, if traders consider non-myopic strategies, *this set of strategies is not an equilibrium*. The exact position of the points does not matter; all we need is the relative position, and the observation that, because of the perfect symmetry in the setup, *segments XY, BC, and AD* are all parallel to each other. Now, suppose trader 1 got a 0. He could move to X instead of Y, to mislead trader 2 into thinking he got a 1. Then, when trader 2 moved to, say, D, trader 1 could correct the rating to A. To show that this is a profitable deviation, observe that this strategy is equivalent to playing two additional moves over trader 1's myopic strategy of moving to Y. The first move, YX may either move toward or away from the optimal final position. The second move, DA or BC, is always in the correct direction. Further, because DA and BC are longer than XY, and parallel to XY, their projection on the final p-line will always be greater in absolute value than the projection of XY, regardless of what the true p-line is! Thus, the deception would result in a strictly higher expected profit for trader 1. Note that this problem is not specific to the projection market form: Our equivalence results show that it could arise in the MSR or DPM (perhaps with a different prior distribution and different numerical values). Observe also that a strategy profile in which neither trader moved in the first two rounds, and trader 1 moved to either X or Y would be a subgame-perfect equilibrium in this setup.

We suggest that one approach to mitigating this problem might be by *reducing* the radius at every move. This essentially provides a form of discounting that motivates trader 1 to take his profit early rather than mislead trader 2. Graphically, the right reduction factor would make the segments AD and BC shorter than XY (as they are chords on a smaller circle), thus making the myopic strategy optimal. This approach has been subsequently generalized by Dimitrov and Sami [32].

This example highlights the advantages of working with the projection market. While it certainly possible to prove nonexistence of myopic equilibrium directly for an example in the MSR or DPM setting, the projection market view enabled us to construct an example and argue that the myopic strategy is not an equilibrium geometrically, *without a single equation*! Further, the geometric insight naturally leads to an idea for a way to solve this problem.

8.7 Concluding remarks on markets with a market maker

We have presented a simple geometric market, the projection market, that can serve as a model for strategic behavior in information markets, as well as a tool to guide the design of new information markets. We have used this model to analyze the cost, profit, and strategies of a trader in a dynamic parimutuel market, and shown that both the dynamic parimutuel market and the spherical market scoring rule are strategically equivalent to the restricted projection market under slight distortion of the prior probabilities.

The general analysis was based on the assumption that traders do not actively try to mislead other traders for future profit. In section 8.6, however, we analyze a small example market without this assumption. We demonstrate that the projection market can be used to analyze traders' strategies in this scenario, and potentially to help design markets with better strategic properties.

Our results raise several very interesting open questions. Firstly, the payoffs of the unrestricted projection market cannot be directly implemented in situations in which the true probability is not ultimately revealed. It would be very useful to have an automatic transformation of a given projection market into another market in which the payoffs can be implemented in expectation without knowing the probability, and preserves the strategic properties of the projection market. Second, given the tight connection between the projection market and the spherical market scoring rule, it is natural to ask if we can find as strong a connection to other scoring rules (perhaps by restricting the projection market to a surface other than the sphere), or if not, to understand what strategic differences are implied by the form of the scoring rule used in the market. Finally, the existence of long-range manipulative strategies in information markets is of great interest. The example we studied in section 8.6 merely scratches the surface of this area. A general study of this class of manipulations, together with a characterization of markets in which it can or cannot arise, would be very useful for the design of information markets.

CHAPTER 9

Prediction markets with an auctioneer

In the previous chapter we examined prediction markets with a market maker, who sets a price for each security at any given time based on the investments in the market so far, and guarantees to fulfill all trader orders at that price. Alternatively, prediction market designs can be set up as an auction, similarly to stock markets, where traders submit orders with prices and a book-keeper or *auctioneer* risklessly finds matching orders. In contrast to the market maker who may incur a loss, the auctioneer does not bear any risk and only matches orders.

In this chapter we focus on the design and computational challenges of auction-based prediction markets for events with a complex (exponentially large) outcome space such as the ranking of n candidates or teams in a tournament.

Almost all existing financial and betting exchanges pair up bilateral trading partners. For example, one trader willing to accept an x dollar loss if a Democrat does not win in return for a y dollar profit if a Democrat wins is matched up with a second trader willing to accept the opposite. However, in many scenarios, even if no bilateral agreements exist among traders, multilateral agreements may be possible. For example, if one trader bets that the Democratic candidate will receive more votes than the Republican candidate, a second trader bets that the Republican candidate will receive more votes than the Libertarian candidate, and a third trader bets that the Libertarian candidate will receive more votes than the Democratic candidate, then, depending on the odds they each offer, there may be a three-way agreeable match even though no two-way matches exist.

In this chapter we propose an exchange where traders have considerable flexibility to naturally and succinctly express their wagers, and examine the computational complexity of the auctioneer's resulting *matching problem* of identifying bilateral and multilateral agreements. For example, suppose that there are n candidates in an election (or n horses in a race, etc.) and thus $n!$ possible orderings of candidates after the final vote tally. Traders may like to bet on arbitrary properties of the final ordering: for example, "candidate D will win", "candidate D will finish in either first place or last place", "candidate D will defeat candidate R", "candidates D and R will both defeat candidate L", etc. The goal of the exchange is to search among all the offers to find two or more that together form an agreeable match. As we shall see, the matching problem can be set up as a linear or integer program, depending on whether orders are divisible or indivisible, respectively. Attempting to reduce the problem to a bilateral matching problem by explicitly creating $n!$ securities, one for each possible final ordering, is both cumbersome for the traders and computationally infeasible even for modest sized n. Moreover, traders' attention would be spread among $n!$ independent choices, making the likelihood of two traders converging at the same time and place seem remote.

There is a trade-off between the expressiveness of the bidding language and the computational complexity of the matching problem. We want to offer traders the most expressive bidding language possible while maintaining computational feasibility. We explore two bidding languages that seem natural from a trader perspective. *Subset betting*, described in Section 9.2.2, allows traders to bet on which positions in the ranking a candidate will fall, for example, "candidate D will finish in position 1, 3-5, or 10". Symmetrically, traders can also bet on which candidates will fall in a particular position. In Section 9.3, we derive a polynomial-time algorithm for matching (divisible) subset bets. The key to the result is showing that the exponentially big linear program has a corresponding separation problem that reduces to maximum weighted bipartite matching and consequently we can solve it in time polynomial in the number of orders.

Pair betting, described in Section 9.2.3, allows traders to bet on the final ranking of any two candidates, for example, "candidate D will defeat candidate R". In Section 9.4, we show that optimal matching of (divisible or indivisible) pair bets is NP-hard, via a reduction from the unweighted minimum feedback arc set problem, which we describe below. We also provide a polynomially-verifiable sufficient condition for the existence of a pair-betting match and show that a greedy algorithm offers poor approximation for indivisible pair bets.

9.1 Design goal and challenges

In a typical horse race, people bet on properties of the outcome like "horse A will win", "horse A will *show*, or finish in either first or second place", or "horses A and B will finish in first and second place, respectively". In practice at the racetrack, each of these different types of bets are processed in separate *pools* or groups. In other words, all the "win" bets are processed together, and all the "show" bets are processed together, but the two types of bets do not mix. This separation can hurt liquidity and information aggregation. For example, even though horse A is heavily favored to win, that may not directly boost the horse's odds to show.

Instead, we describe a central exchange where all bets on the outcome are processed together, thus aggregating liquidity and ensuring that informational inference happens automatically. Ideally, we'd like to allow traders to bet on any property of the final ordering they like, stated in exactly the language they prefer. In practice, allowing too flexible a language creates a computational burden for the auctioneer attempting to match willing traders. We explore the trade-off between the expressiveness of the bidding language and the computational complexity of the matching problem.

We consider a framework where people propose to buy securities that pay \$1 if and only if some property of the final ordering is true. Traders state the price they are willing to pay per share and the number of shares they would like to purchase. (Sell orders may not be explicitly needed, since buying the negation of an event is equivalent to selling the event.) A *divisible* order permits the trader to receive fewer shares than requested, as long as the price constraint is met; an *indivisible* order is an all-or-nothing order. The description of bets in terms of prices and shares is without loss of generality: we can also allow bets to be described in terms of odds, payoff vectors, or any of the diverse array of approaches practiced in financial and gambling circles.

In principle, we can do everything we want by explicitly offering $n!$ securities, one for every state $s \in \mathcal{S}$ (or in fact any set of $n!$ linearly independent securities). This is the so-called *complete Arrow-Debreu securities market* [4] for our setting. In practice, traders do not want to deal with low-level specification of complete orderings: people think more naturally in terms of high-level properties of orderings. Moreover, operating $n!$ securities is infeasible in practice from a computational point of view as n grows.

A very simple bidding language might allow traders to bet only on who wins the competition, as is done in the "win" pool at racetracks. The corresponding matching problem is polynomial; however, the language is not very expressive. A trader who believes that A will defeat B, but that neither will win outright cannot usefully impart his information to the market. The price space of the market reveals the collective estimates of win probabilities but nothing else. *Our goal is to find languages that are as expressive and intuitive as possible and reveal as much information as possible, while maintaining computational feasibility.*

Our presentation on the design of bidding languages vs the complexity of the auctioneer's matching problem in the context of permutation-type outcomes is in direct analogy to work by Fortnow et. al. [45], who consider the 2^n possible outcomes of a logical formula with n binary variables. Traders express bets in Boolean logic. The authors show that divisible matching is co-NP-complete and indivisible matching is Σ_2^p-complete.

Research on bidding languages and winner determination in combinatorial auctions [27, 101, 118] considers similar computational challenges in finding an allocation of items to bidders that maximizes the auctioneer's revenue. Combinatorial auctions allow bidders to place distinct values on bundles of goods rather than just on individual goods. Uncertainty and risk are typically not considered and the central auctioneer problem is to maximize social welfare. Our mechanisms allow traders to construct bets for an event with $n!$ outcomes. *Uncertainty and risk* are considered and the auctioneer problem is to explore arbitrage opportunities and risklessly match up wagers.

9.2 Permutation betting

In this section, we define the matching and optimal matching problems that an auctioneer needs to solve in a general permutation betting market. We then illustrate the problem definitions in the context of the subset-betting and pair-betting markets.

9.2.1 Securities, orders and matching problems

Consider an event with n competing candidates where the outcome (state) is a ranking of the n candidates. The bidding language of a market offering securities in the future outcomes determines the type and number of securities available and directly affects what information can be aggregated about the outcome. A fully expressive bidding language can capture any possible information that traders may have about the final ranking; a less expressive language limits the type of information that can be aggregated though it may enable a more efficient solution to the matching problem. For any bidding language and number of securities in a permutation betting market, we can succinctly represent the problem of the auctioneer to risklessly match offers as follows.

Consider an index set of bets or orders $\mathcal{O}$ which traders submit to the auctioneer. Each order $i \in \mathcal{O}$ is a triple (b_i, q_i, ϕ_i), where b_i denotes how much the trader is willing to pay for a unit share of security ϕ_i and q_i is the number of shares of the security he wants to purchase at price b_i. Naturally, $b_i \in (0, 1)$ since a unit of the security pays off at most \$1 when the event is realized. Since order i is defined for a single security ϕ_i, we will omit the security variable whenever it is clear from the context.

The auctioneer can accept or reject each order, or in a divisible world accept a fraction of the order. Let x_i be the fraction of order $i \in \mathcal{O}$ accepted. In the indivisible version of the market $x_i = 0$ or 1 while in the divisible version $x_i \in [0, 1]$. Further let $I_i(s)$ be the indicator variable for whether order i is winning in state s, that is $I_i(s) = 1$ if the order is paid back \$1 in state s and $I_i(s) = 0$ otherwise.

There are two possible problems that the auctioneer may want to solve. The simpler one is to find a subset of orders that can be matched risk-free, namely a subset of orders which accepted together give a nonnegative profit to the auctioneer in every possible outcome. We call this problem the *existence of a match* or sometimes simply, the matching problem. The more complex problem is for the auctioneer to find the *optimal match* with respect to some criterion such as profit, trading volume, etc.

Definition 9.2.1 (Existence of match, indivisible orders). *Given a set of orders $\mathcal{O}$, does there exist a set of $x_i \in \{0, 1\}$, $i \in \mathcal{O}$, with at least one $x_i = 1$ such that*

$$\sum_i (b_i - I_i(s))q_i x_i \geq 0, \quad \forall s \in \mathcal{S}? \tag{9.1}$$

Similarly we can define the existence of a match with divisible orders.

Definition 9.2.2 (Existence of match, divisible orders). *Given a set of orders $\mathcal{O}$, does there exist a set of $x_i \in [0, 1]$, $i \in \mathcal{O}$, with at least one $x_i > 0$ such that*

$$\sum_i (b_i - I_i(s))q_i x_i \geq 0, \quad \forall s \in \mathcal{S}? \tag{9.2}$$

The existence of a match is a decision problem. It only returns whether trade can occur at no risk to the auctioneer. In addition to the risk-free requirement, the auctioneer can optimize some criterion in determining the orders to accept. Some reasonable objectives include maximizing the total trading volume in the market or the worst-case profit of the auctioneer. The following optimal matching problems are defined for an auctioneer who maximizes his worst-case profit.

Definition 9.2.3 (Optimal match, indivisible orders). *Given a set of orders $\mathcal{O}$, choose $x_i \in \{0, 1\}$ such that the following mixed integer programming problem achieves its optimality*

$$\begin{aligned} \max_{x_i, c} \quad & c \\ \text{s.t.} \quad & \textstyle\sum_i \big(b_i - I_i(s)\big)q_i x_i \geq c, \quad \forall s \in \mathcal{S} \\ & x_i \in \{0, 1\}, \qquad \forall i \in \mathcal{O}. \end{aligned} \tag{9.3}$$

Definition 9.2.4 (Optimal match, divisible orders). *Given a set of orders $\mathcal{O}$, choose $x_i \in [0,1]$ such that the following linear programming problem achieves its optimality*

$$\begin{aligned} \max_{x_i, c} \quad & c \\ \text{s.t.} \quad & \textstyle\sum_i \big(b_i - I_i(s)\big) q_i x_i \geq c, \quad \forall s \in \mathcal{S} \\ & 0 \leq x_i \leq 1, \qquad\qquad\quad \forall i \in \mathcal{O}. \end{aligned} \tag{9.4}$$

The variable c is the worst-case profit for the auctioneer. Note that, strictly speaking, the optimal matching problems do not require to solve the optimization problems (9.3) and (9.4), because only the optimal set of orders are needed. The optimal worst-case profit may remain unknown.

9.2.2 Subset betting

A subset betting market allows two different types of bets. Traders can bet on a subset of positions a candidate may end up at, or they can bet on a subset of candidates that will occupy a particular position. A security $\langle \alpha | \Phi \rangle$ where Φ is a subset of positions pays off \$1 if candidate α stands at a position that is an element of Φ and it pays \$0 otherwise. For example, security $\langle \alpha | \{2, 4\} \rangle$ pays \$1 when candidate α is ranked second or fourth. Similarly, a security $\langle \Psi | j \rangle$ where Ψ is a subset of candidates pays off \$1 if any of the candidates in the set Ψ ranks at position j. For instance, security $\langle \{\alpha, \gamma\} | 2 \rangle$ pays off \$1 when either candidate α or candidate γ is ranked second.

The auctioneer in a subset betting market faces a non-trivial matching problem, that is to determine which orders to accept among all submitted orders $i \in \mathcal{O}$. Note that although there are only n candidates and n possible positions, the number of available securities to bet on is exponential since a trader may bet on any of the 2^n subsets of candidates or positions. With this, it is not immediately clear whether one can even find a trading partner or a match for trade to occur, or that the auctioneer can solve the matching problem in polynomial time. In the next section, we will show that somewhat surprisingly there is an elegant polynomial solution to both the matching and optimal matching problems, based on classic combinatorial problems.

When an order is accepted, the corresponding trader pays the submitted order price b_i to the auctioneer and the auctioneer pays the winning orders \$1 per share after the outcome is revealed. The auctioneer has to carefully choose which orders and what fractions of them to accept so as to be guaranteed a nonnegative profit in any future state. The following example illustrates the matching problem for indivisible orders in the subset-betting market.

Example 9.2.5. *Suppose $n = 3$. Objects α, β, and γ compete for positions 1, 2, and 3 in a competition. The auctioneer receives the following 4 orders: (1) buy 1 share $\langle \alpha | \{1\} \rangle$ at price \$0.6; (2) buy 1 share $\langle \beta | \{1, 2\} \rangle$ at price \$0.7; (3) buy 1 share $\langle \gamma | \{1, 3\} \rangle$ at price \$0.8; and (4) buy 1 share $\langle \beta | \{3\} \rangle$ at price \$0.7. There are 6 possible states of ordering: $\alpha\beta\gamma$, $\alpha\gamma\beta$, $\beta\alpha\gamma$, $\beta\gamma\alpha$, $\gamma\alpha\beta$, and $\gamma\beta\alpha$. The corresponding state-dependent profit of the auctioneer for each order can be* *calculated as the following vectors,*

$$\begin{aligned}
c_1 &= (-0.4, -0.4,\ \ 0.6,\ \ 0.6,\ \ 0.6,\ \ 0.6)\\
c_2 &= (-0.3,\ \ 0.7, -0.3, -0.3,\ \ 0.7, -0.3)\\
c_3 &= (-0.2,\ \ 0.8, -0.2,\ \ 0.8, -0.2, -0.2)\\
c_4 &= (\ \ 0.7, -0.3,\ \ 0.7,\ \ 0.7, -0.3,\ \ 0.7).
\end{aligned}$$

6 columns correspond to the 6 future states. For indivisible orders, the auctioneer can either accept orders (2) and (4) and obtain profit vector

$$c = (0.4,\quad 0.4,\quad 0.4,\quad 0.4,\quad 0.4,\quad 0.4),$$

or accept orders (2), (3), and (4) and has profit across state

$$c = (0.2,\quad 1.2,\quad 0.2,\quad 1.2,\quad 0.2,\quad 0.2).$$

9.2.3 Pair betting

A pair betting market allows traders to bet on whether one candidate will rank higher than another candidate, in an outcome which is a permutation of n candidates. A security $\langle \alpha > \beta \rangle$ pays off \$ 1 if candidate α is ranked higher than candidate β and \$ 0 otherwise. There are a total of $N(N-1)$ different securities offered in the market, each corresponding to an ordered pair of candidates.

Traders place orders of the form "buy q_i shares of $\langle \alpha > \beta \rangle$ at price per share no greater than b_i". b_i in general should be between 0 and 1. Again the order can be either indivisible or divisible and the auctioneer needs to decide what fraction x_i of each order to accept so as not to incur any loss, with $x_i \in \{0, 1\}$ for indivisible and $x_i \in [0, 1]$ for divisible orders. The same definitions for existence of a match and optimal match from Section 9.2.1 apply.

The orders in the pair-betting market have a natural interpretation as a graph, where the candidates are nodes in the graph and each order which ranks a pair of candidates $\alpha > \beta$ is represented by a directed edge $e = (\alpha, \beta)$ with price b_e and weight q_e. With this interpretation, it is tempting to assume that a necessary condition for a match is to have a cycle in the graph with a nonnegative worst-case profit. Assuming $q_e = 1$ for all e, this is a cycle C with a total of $|C|$ edges such that the worst-case profit for the auctioneer is

$$\left(\sum_{e \in C} b_e\right) - (|C| - 1) \geq 0,$$

since in the worst-case state the auctioneer needs to pay \$,1 to every order in the cycle except one. However, the example in Figure 9.1 shows that this is not the case: we may have a set of orders in which every single cycle has a negative worst-case profit, and yet there is a positive worst-case match overall. The edge labels in the figure are the prices b_e; both the optimal divisible and indivisible solution in this case accept all orders in full, $x_e = 1$.

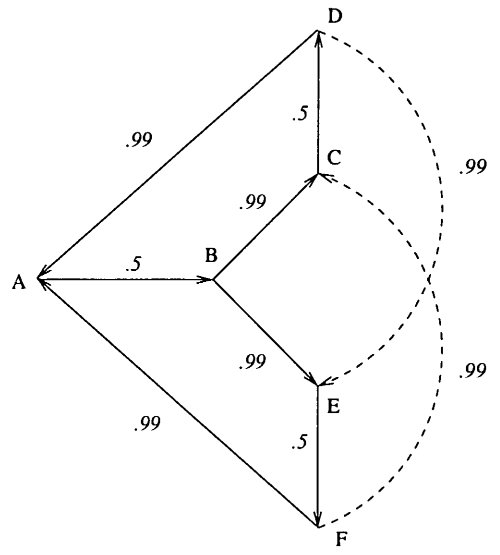


Figure 9.1. Every cycle has negative worst-case profit of -0.02 (for the cycles of length 4) or less (for the cycles of length 6); however, accepting all edges in full gives a positive worst-case profit of 0.44.

9.3 Complexity of subset betting

The matching problems of the auctioneer in any permutation market, including the subset betting market have $n!$ constraints. Brute-force methods would take exponential time to solve. However, given the special form of the securities in the subset betting market, we can show that the matching problems for divisible orders can be solved in polynomial time.

Theorem 9.3.1. *The existence of a match and the optimal match problems with divisible orders in a subset betting market can both be solved in polynomial time.*

Proof. Consider the linear programming problem (9.4) for finding an optimal match. This linear program has $|\mathcal{O}| + 1$ variables, one variable x_i for each order i and the profit variable c. It also has exponentially many constraints. However, we can solve the program in time polynomial in the number of orders $|\mathcal{O}|$ by using the ellipsoid algorithm, as long as we can efficiently solve its corresponding *separation* problem in polynomial time [57, 58]. The separation problem for a linear program takes as input a vector of variable values and returns if the vector is feasible, or otherwise it returns a violated constraint.

For given values of the variables, a violated constraint in Eq. (9.4) asks whether there is a state or permutation s in which the profit is less than c, and can be rewritten as

$$\sum_i I_i(s) q_i x_i < \Big(\sum_i b_i q_i x_i\Big) - c \quad \forall s \in \mathcal{S}. \tag{9.5}$$

Thus it suffices to show how to find efficiently a state s satisfying the above inequality (9.5) or verify that the opposite inequality holds for all states s.

We will show that the separation problem can be reduced to the *maximum weighted bipartite* *matching*[1] problem [26]. The left hand side in Eq. (9.5) is the total money that the auctioneer needs to pay back to the winning traders in state s. The first term on the right hand side is the total money collected by the auctioneer and it is fixed for a given solution vector of x_i's and c. A weighted bipartite graph can be constructed between the set of candidates and the set of positions. For every order of the form $\langle \alpha | \Phi \rangle$ there are edges from candidate node α to every position node in Φ. For orders of the form $\langle \Psi | j \rangle$ there are edges from each candidate in Ψ to position j. For each order i we put weight $q_i x_i$ on each of these edges. All multi-edges with the same end points are then replaced with a single edge that carries the total weight of the multi-edge. Every state s then corresponds to a perfect matching in the bipartite graph. In addition, the auctioneer pays out to the winners the sum of all edge weights in the perfect matching since every candidate can only stand in one position and every position is taken by one candidate. Thus, the auctioneer's worst-cast state and payment are the solution to the maximum weighted bipartite matching problem, which has known polynomial-time algorithms [82, 95]. Hence, the separation problem can be solved in polynomial time.

Naturally, if the optimal solution to (9.4) gives a worst-case profit of $c^* > 0$, there exists a matching. Thus, the matching problem can be solved in polynomial time also. □

■ 9.4 Complexity of pair betting

In this section we show that a slight change of the bidding language may bring about a dramatic change in the complexity of the optimal matching problem of the auctioneer. In particular, we show that finding the optimal match in the pair betting market is NP-hard for both divisible and indivisible orders. We then identify a polynomially-verifiable sufficient condition for deciding the existence of a match.

The hardness results are surprising especially in light of the observation that a pair betting market has a seemingly more restrictive bidding language which only offers $n(n-1)$ securities. In contrast, the subset betting market enables traders to bet on an exponential number of securities and yet had a polynomial time solution for finding the optimal match. Our hope is that the comparison of the complexities of the subset and pair betting markets would offer insight into what makes a bidding language expressive while at the same time enabling an efficient matching solution.

In all analysis that follows, we assume that traders submit unit orders in pair betting markets, that is $q_i = 1$. A set of orders $\mathcal{O}$ received by the auctioneer in a pair betting market with unit orders can be represented by a directed graph, $G(V, E)$, where the vertex set V contains candidates that traders bet on. An edge $e \in E$, denoted (α, β, b_e), represents an order to buy 1 share of the security $\langle \alpha > \beta \rangle$ at price b_e. All edges have equal weight of 1.

We will repeatedly use the minimum feedback arc set problem in our complexity analysis on pair betting markets, which is defined as follows. A *feedback arc set* of a directed graph is a set of edges which, when removed from the graph, leave a directed acyclic graph (DAG). *Unweighted minimum feedback arc set* problem is to find a feedback arc set with the minimum cardinality, while *weighted minimum feedback arc set* problem seeks to find a feedback arc set with the minimum

[1]The notion of perfect matching in a bipartite graph, which we use only in this proof, should not be confused with the notion of matching bets which we use throughout this chapter.

total edge weight. Both unweighted and weighted minimum feedback arc set problems have been shown to be NP-complete [76].

We adopt the following notations:

- $G(V, E)$: original equally weighted directed graph for the set of unit orders $\mathcal{O}$.
- b_e: price of the order for edge e.
- $G^*(V^*, E^*)$; a weighted directed graph of accepted orders for optimal matching, where edge weight x_e is the quantity of order e accepted by the auctioneer. $x_e = 1$ for indivisible orders and $0 < x_e \leq 1$ for divisible orders.
- $H(V, E)$: a generic weighted directed graph of accepted orders.
- $k(H)$: solution to the unweighted minimum feedback arc set problem on graph H. $k(H)$ is the minimum number of edges to remove so that H becomes acyclic.
- $l(H)$: solution to the weighted minimum feedback arc set problem on graph H. $l(H)$ is the minimum total weights for the set of edges which, when removed, leave H acyclic.
- $c(H)$: worst-case profit of the auctioneer if he accepts all orders in graph H.
- ϵ: a sufficiently small positive real number. Where not stated, $\epsilon < 1/(2|E|)$ for a graph $H(V, E)$. In other cases, the value is determined in context.

9.4.1 Optimal indivisible matching

The auctioneer's optimal indivisible matching problem is introduced in Definition 9.2.3 of Section 9.2. Assuming unit orders and considering the order graph $G(V, E)$, we restate the auctioneer's optimal matching problem in a pair betting market as picking a subset of edges to accept such that worst-case profit is maximized in the following optimization problem,

$$\begin{aligned} \max_{x_e, c} \quad & c \\ \text{s.t.} \quad & \textstyle\sum_e \big(b_e - I_e(s)\big)x_e \geq c, \quad \forall s \in \mathcal{S} \\ & x_e \in \{0, 1\}, \qquad \forall e \in E. \end{aligned} \tag{9.6}$$

Without loss of generality, we assume that there are no multi-edges in the order graph G.

We show that the optimal matching problem for indivisible orders is NP-hard via a reduction from the unweighted minimum feedback arc set problem. The latter takes as input a directed graph, and asks what is the minimum number of edges to delete from the graph so as to be left with a DAG. Our hardness proof is based on the following lemmas.

Lemma 9.4.1. *Suppose the auctioneer accepts all edges in an equally weighted directed graph $H(V, E)$ with edge price $b_e = (1 - \epsilon)$ and edge weight $x_e = 1$. Then the worst-case profit is equal to $k(H) - \epsilon|E|$, where $k(H)$ is the solution to the unweighted minimum feedback arc problem on H.*

Proof. If the order of an edge gets \$1 payoff at the end of the market we call the edge a winning edge, otherwise it is called a losing edge. For any state s, all winning edges necessarily form a DAG. Conversely, for every DAG there is a state in which the DAG edges are winners (though the remaining edges in G are not necessarily losers).

Suppose that in state s there are w_s winning edges and $l_s = |E| - w_s$ losing edges. Then, l_s is the cardinality of a feedback arc set that consists of all losing edges in state s. The edges that remain after deleting the minimum feedback arc set form the maximum DAG for the graph H. Consider the state s_{max} in which all edges of the maximum DAG are winners. This gives the maximum number of winning edges w_{max}. All other edges are necessarily losers in the state s_{max}, since any edge which is not in the max DAG must form a cycle together with some of the DAG edges. The number of losing edges in state s_{max} is the cardinality of the minimum feedback arc set of H, that is $|E| - w_{max} = k(H)$.

The profit of the auctioneer in a state s is

$$\begin{aligned} profit(s) &= \Big(\sum_{e \in E} b_e\Big) - w \\ &= (1-\epsilon)|E| - w \\ &\geq (1-\epsilon)|E| - w_{max}, \end{aligned}$$

where equality holds when $s = s_{max}$. Thus, the worst-case profit is achieved at state s_{max},

$$profit(s_{max}) = (|E| - w_{max}) - \epsilon|E| = k(H) - \epsilon|E|.$$

□

Consider the graph of accepted orders for optimal matching, $G^*(V^*, E^*)$, which consists of the optimal subset of edges E^* to be accepted by the auctioneer, that is edges with $x_e = 1$ in the solution of the optimization problem (9.6). We have the following lemma.

Lemma 9.4.2. *If the edge prices are $b_e = (1 - \epsilon)$, then the optimal indivisible solution graph G^* has the same unweighted minimum feedback arc set size as the graph of all orders G, that is $k(G^*) = k(G)$. Furthermore, G^* is the smallest such subgraph of G, i.e., it is the subgraph of G with the smallest number of edges, that has the same size of unweighted minimum feedback arc set as G.*

Proof. G^* is a subgraph of G, hence the minimum number of edges to break cycles in G^* is no more than that in G, namely $k(G^*) \leq k(G)$.

Suppose $k(G^*) < k(G)$. Since both $k(G^*)$ and $k(G)$ are integers, for any $\epsilon < \frac{1}{|E|}$ we have that $k(G^*) - \epsilon|E^*| < k(G) - \epsilon|E|$. Hence by Lemma 9.4.1, the auctioneer has a higher worst-case profit by accepting G than accepting G^*, which contradicts the optimality of G^*. Finally, the worst-case profit $k(G) - \epsilon|E^*|$ is maximized when $|E^*|$ is minimized. Hence, G^* is the smallest subgraph of G such that $k(G^*) = k(G)$. □

The above two lemmas prove that the maximum worst-case profit in the optimal indivisible matching is directly related to the size of the minimum feedback arc set. Thus computing each automatically gives the other, hence computing the maximum worst-case profit in the indivisible pair betting problem is NP-hard.

Theorem 9.4.3. *Computing the maximum worst-case profit in indivisible pair betting is NP-hard.*

Proof. By Lemma 9.4.2, the maximum worst-case profit which is the optimum to the mixed integer programming problem (9.6), is $k(G) - \epsilon|E^*|$, where $|E^*|$ is the number of accepted edges. Since $k(G)$ is integer and $\epsilon|E^*| \leq \epsilon|E| < 1$, solving (9.6) will automatically give us the cardinality of the minimum feedback arc set of G, $k(G)$. Because the minimum feedback arc set problem is NP-complete [76], computing the maximum worst-case profit is NP-hard. □

Theorem 9.4.3 states that solving the optimization problem is hard, because even if the optimal set of orders are provided computing the optimal worst-case profit from accepting those orders is NP-hard. However, it does not imply whether the optimal matching problem, i.e. finding the optimal set of orders to accept, is NP-hard. It is possible to be able to determine which edges in a graph participating in the optimal match, yet unable to compute the corresponding worst-case profit. Next, we prove that the indivisible optimal matching problem is actually NP-hard. We will use the following short fact repeatedly.

Lemma 9.4.4 (Edge removal lemma). *Given a weighted graph $H(V, E)$, removing a single edge e with weight x_e from the graph decreases the weighted minimum feedback arc set solution $l(H)$ by no more than x_e and reduces the unweighted minimum feedback arc set solution $k(H)$ by no more than* 1.

Proof. Suppose the weighted minimum feedback arc set for the graph $H-\{e\}$ is F. Then $F\cup\{e\}$ is a feedback arc set for H, and has total edge weight $l(H - \{e\}) + x_e$. Because $l(H)$ is the solution to the weighted minimum feedback arc set problem on H, we have $l(H) \leq l(H - \{e\}) + x_e$, implying that $l(H - \{e\}) \geq l(H) - x_e$.

Similarly, suppose the unweighted minimum feedback arc set for the graph $H - \{e\}$ is F'. Then $F' \cup \{e\}$ is a feedback arc set for H, and has set cardinality $k(H - \{e\}) + 1$. Because $k(H)$ is the solution to the unweighted minimum feedback arc set problem on H, we have $k(H) \leq k(H - \{e\}) + 1$, giving that $k(H - \{e\}) \geq k(H) - 1$. □

Theorem 9.4.5. *Finding the optimal match in indivisible pair betting is NP-hard.*

Proof. We reduce from the unweighted minimum feedback arc set problem again, although with a slightly more complex polynomial transformation involving multiple calls to the optimal match oracle. Consider an instance graph G of the minimum feedback arc set problem. We are interested in computing $k(G)$, the size of the minimum feedback arc set of G.

Suppose we have an oracle which solves the optimal matching problem.
Denote by $optimal\_match(G')$ the output of the optimal matching oracle on graph G' with prices $b_e = (1 - \epsilon)$ on all its edges. By Lemma 9.4.2, on input G', the oracle $optimal\_match$ returns the subgraph of G' with the smallest number of edges, that has the same size of minimum feedback arc set as G'.

The following procedure finds $k(G)$ by using polynomially many calls to the optimal match oracle on a sequence of subgraphs of G.

This procedure removes edges from the original graph G layer by layer until the graph is empty, while at the same time computing the minimum feedback arc set size $k(G)$ of the original graph as

```
set $G' := G$
iterations $:= 0$
while ($G'$ has nonempty edge set)
    reset $G' := optimal\_match(G')$
    if ($G'$ has nonempty edge set)
        increment iterations by 1
        reset $G'$ by removing any edge $e$
    end if
end while
return (iterations)
```

the number of iterations. In each iteration, we start with a graph G' and replace it with the smallest subgraph G' that has the same $k(G')$. At this stage, removing an additional edge e necessarily results in $k(G' - \{e\}) = k(G') - 1$, because $k(G' - \{e\}) < k(G')$ by the optimality of G', and $k(G' - \{e\}) \geq k(G') - 1$ by the edge-removal lemma. Therefore, in each iteration the cardinality of the minimum feedback arc set gets reduced exactly by 1. Hence the number of iterations is equal to $k(G)$.

Note that this procedure gives a polynomial transformation from the optimal matching problem to the unweighted minimum feedback arc set problem, which calls the optimal matching oracle exactly $k(G) \leq |E|$ times, where $|E|$ is the number of edges of G. Hence the optimal matching problem is NP-hard. □

9.4.2 Optimal divisible matching

When orders are divisible, the auctioneer's optimal matching problem is described in Definition 9.2.4 of Section 9.2. Assuming unit orders and considering the order graph $G(V, E)$, we restate the auctioneer's optimal matching problem for divisible orders as choosing quantity of orders to accept, $x_e \in [0, 1]$, such that worst-case profit is maximized in the following linear programming problem,

$$\begin{aligned} \max_{x_e, c} \quad & c \\ \text{s.t.} \quad & \textstyle\sum_e \big(b_e - I_e(s)\big)x_e \geq c, \quad \forall s \in \mathcal{S} \\ & x_e \in [0, 1], \qquad \forall e \in E. \end{aligned} \tag{9.7}$$

We still assume that there are no multi-edges in the order graph G.

When orders are divisible, the auctioneer can be better off by accepting partial orders. Example 9.4.6 shows a situation when accepting partial orders generates higher worst-case profit than the optimal indivisible solution.

Example 9.4.6. *We show that the linear program (9.7) sometimes has a non-integer optimal solution.*

Consider the graph in Figure 9.2. There are a total of five cycles in the graph: three four-edge cycles ABCD, ABEF, CDEF, and two six-edge cycles ABCDEF and ABEFCD. Suppose each edge

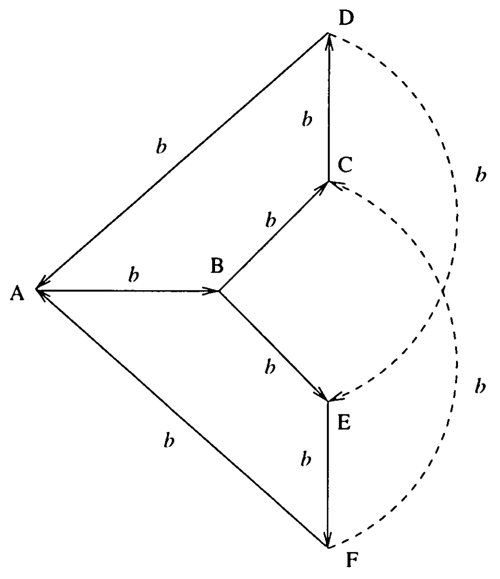


Figure 9.2. An order graph. Letters on edges represent order prices.

has price b such that $4b - 3 > 0$ and $6b - 5 < 0$, namely $b \in (.75, .80)$, for example, $b = .78$. With this, the optimal indivisible solution consists of at most one four-edge cycle, with worst case profit $(4b - 3)$. On the other hand, taking $\frac{1}{2}$ fraction of each of the three four-edge cycles would yield higher worst-case profit of $\frac{3}{2}(4b - 3)$.

Despite the potential profit increase for accepting divisible orders, the auctioneer's optimal matching problem remains to be NP-hard for divisible orders, which is presented below via several lemmas and theorems.

Lemma 9.4.7. *Suppose the auctioneer accept orders described by a weighted directed graph $H(V, E)$ with edge weight x_e to be the quantity accepted for edge order e. The worst-case profit for the auctioneer is*

$$c(H) = \sum_{e \in E}(b_e - 1)x_e + l(H). \tag{9.8}$$

Proof. For any state s, the winning edges form a DAG. Thus, the worst-case profit for the auctioneer achieves at the state(s) when the total quantity of losing orders is minimized. The minimum total quantity of losing orders is the solution to weighted minimal feedback arc set problem on H, that is $l(H)$. □

Consider the graph of accepted orders for optimal divisible matching, $G^*(V^*, E^*)$, which consists of the optimal subset of edges E^* to be accepted by the auctioneer, with edge weight $x_e > 0$ getting from the optimal solution of the linear program (9.7). We have the following lemmas.

Lemma 9.4.8. *$l(G^*) \leq k(G^*) \leq k(G)$.*

Proof. $l(G^*)$ is the solution of the weighted minimum feedback arc set problem on G^*, while $k(G^*)$ is the solution of the unweighted minimum feedback arc set problem on G^*. When all edge weights in G^* are 1, $l(G^*) = k(G^*)$. When x_e's are less than 1, $l(G^*)$ can be less than or equal to $k(G^*)$. Since G^* is a subgraph of G but possibly with different edge weights, $k(G^*) \leq k(G)$. Hence, we have the above relation. □

Lemma 9.4.9. *If $\epsilon < 1/(2|E|)$ and all edge prices b_e's are $(1-\epsilon)$, then $k(G^*) = k(G)$ and $k(G^*) - 1/2 < l(G^*) \leq k(G^*)$.*

Proof. We know that the auctioneer's worst-case profit when accepting G^* is

$$c(G^*) = \sum_{e \in E^*} (b_e - 1)x_e + l(G^*) = l(G^*) - \epsilon \sum_{e \in E^*} x_e.$$

When he accepts the original order graph G in full, his worst-case profit is

$$c(G) = \sum_{e \in E} (b_e - 1) + k(G) = k(G) - \epsilon|E|.$$

Because G^* is optimal, $c(G^*) \geq c(G)$. Thus,

$$l(G^*) - \epsilon \sum_{e \in E^*} x_e \geq k(G) - \epsilon|E|,$$

which gives

$$l(G^*) \geq k(G) - \epsilon|E| + \epsilon \sum_{e \in E^*} x_e \geq k(G) - \epsilon|E| > k(G) - 1/2.$$

According to Lemma 9.4.8,

$$k(G) - 1 < l(G^*) \leq k(G^*) \leq k(G).$$

Since both $k(G)$ and $k(G^*)$ are integers, only when $k(G^*) = k(G)$ the above relations hold. □

Theorem 9.4.10. *Finding the optimal worst-case profit in divisible pair betting is NP-hard.*

Proof. Given the optimal set of partial orders to accept for G when prices b_e's are $(1-\epsilon)$ and $\epsilon < 1/(2|E|)$, if we can calculate the optimal worst-case profit, we can obtain $l(G^*)$. According to Lemma 9.4.9, rounding $l(G^*)$ up to the nearest integer gives the solution to the unweighted minimum feedback arc set problem on G, which is NP-hard. Hence, finding the optimal worst-case profit is NP-hard. □

Theorem 9.4.10 states that solving the linear program (9.7) is NP-hard. Similarly to the indivisible case, we still need to prove that just finding the optimal divisible match is hard, as opposed to being able to compute the optimal worst-case profit. Since in the divisible case the edges do not necessarily have unit weights, the proof in Theorem 9.4.5 does not apply directly. However, with an additional property of the divisible case, we can augment the procedure from the indivisible hardness proof to compute the unweighted minimum feedback arc set size $k(G)$ here as well.

Lemma 9.4.11. *G^* is an optimal divisible graph of G, where $b_e = 1 - \epsilon$ for all edges $e \in E$ and $\epsilon < 1/(2|E|)$. Let e_{min} be an edge with the minimal edge weight in G^*. If $x_{e_{min}} < 1$ and we remove e_{min} from G^*, then $k(G^* - \{e_{min}\}) = k(G)$.*

Proof. Assume the contrary, namely $k(G^* - \{e_{min}\}) < k(G)$. By Lemma 9.4.9, $k(G) = k(G^*)$. By Lemma 9.4.4, $k(G^* - \{e_{min}\}) = k(G^*) - 1$. The auctioneer's worst-case profit when accepting G^* is $l(G^*) - \epsilon \sum_{e \in E^*} x_e$. If the auctioneer accepts G' which is the same as G^* except that all edge weights are 1, his worst-case profit is $k(G^*) - \epsilon|E^*|$. Because G^* is optimal, $l(G^*) - \epsilon \sum_{e \in E^*} x_e \geq k(G^*) - \epsilon|E^*|$, which gives

$$k(G^*) - l(G^*) \leq \epsilon(|E^*| - \sum_{e \in E^*} x_e) \leq \epsilon(|E^*| - |E^*|x_{e_{min}}).$$

Because $\epsilon < 1/(2|E|) <= 1/(2|E^*|)$,

$$k(G^*) - l(G^*) \leq \frac{1}{2}(1 - x_{e_{min}}). \tag{9.9}$$

On the other hand, $G^* - \{e_{min}\} \subset G^*$, so we have

$$l(G^* - \{e_{min}\}) \leq k(G^* - \{e_{min}\}) = k(G^*) - 1. \tag{9.10}$$

Since removing a single edge can not reduce the minimum feedback art set by more than the edge weight,

$$l(G^*) - x_{e_{min}} \leq l(G^* - \{e_{min}\}). \tag{9.11}$$

From (9.10) and (9.11), we have

$$k(G^*) - l(G^*) \geq 1 - x_{e_{min}}. \tag{9.12}$$

Combining (9.9) and (9.12), we get $x_{e_{min}} = 1$, which contradicts $x_{e_{min}} < 1$. Thus, if the minimal edge weight is less than 1, removing the corresponding edge from an optimal divisible graph does not change $k(G)$, the minimal feedback arc set size of the original graph. □

We now can augment the procedure for the indivisible case in Theorem 9.4.5, to prove hardness of the divisible version, as follows.

Theorem 9.4.12. *Finding the optimal match in divisible pair betting is NP-hard.*

Proof. We reduce from the unweighted minimum feedback arc set problem for graph G. Set $b_e = 1 - \epsilon$ for all edges in G, and $\epsilon < 1/(2|E|)$. Suppose we have an oracle for the optimal divisible problem called *optimal_divisible_match*, which on input graph H computes edge weights $x_e \in (0, 1]$ for the optimal subgraph H^* of H. The following procedure outputs $k(G)$.

As in the proof of the corresponding Theorem 9.4.5 for the indivisible case, we compute $k(G)$ by iteratively removing edges and recomputing the optimal divisible solution on the remaining subgraph, until all edges are deleted. In each iteration of the outer while loop, the minimum feedback arc set is reduced by 1, thus the number of iterations is equal to $k(G)$.

```
set $G' := G$
iterations $:= 0$
while ($G'$ has nonempty edge set)
    reset $G' := optimal\_divisible\_match(G')$
    while ($G'$ has edges with weight < 1)
        remove an edge with the minimum weight from $G'$
        reset $G'$ by setting all edge weights to 1
        reset $G' := optimal\_divisible\_match(G')$
    end while
    if ($G'$ has nonempty edge set)
        increment iterations by 1
        reset $G'$ by removing any edge $e$
    end if
end while
return (iterations)
```

It remains to verify that each iteration reduces $k(G)$ by exactly 1. Starting from a graph at the beginning of an iteration, we compute its optimal divisible subgraph. We then keep removing an edge with the minimum weight at a time and recomputing the optimal divisible subgraph, until the latter contains only edges with unit weight. By Lemma 9.4.11 throughout the iteration so far the minimum feedback arc set of the corresponding unweighted graph remains unchanged.

Once the oracle returns a graph G' with only unit edge weights, removing any edge would reduce the minimum feedback arc set: otherwise G' is not optimal since $G' - \{e\}$ would have the same minimum feedback arc set but smaller total edge weight. By Lemma 9.4.4 removing a single edge cannot reduce the minimum feedback arc set by more than one, thus as all edges have unit weight, $k(G')$ gets reduced by exactly one. $k(G)$ is equal to the returned value from the procedure. Hence, the optimal matching problem for divisible orders is NP-hard. □

9.4.3 Existence of a match

Knowing that the optimal matching problem is NP-hard for both indivisible and divisible orders in pair betting, we check whether the auctioneer can identify the existence of a match with ease. Lemma 9.4.13 states a sufficient condition for the matching problem with both indivisible and divisible orders.

Lemma 9.4.13. *A sufficient condition for the existence of a match for pair betting is that there exists a cycle C in G such that,*

$$\sum_{e \in C} b_e \geq |C| - 1, \tag{9.13}$$

where $|C|$ is the number of edges in the cycle C.

Proof. The left-hand side of the inequality (9.13) represents the total payment that the auctioneer receives by accepting every unit orders in the cycle C in full. Because the direction of an edge represents predicted ordering of the two connected nodes in the final ranking, forming a cycle meaning that there is some logical contradiction on the predicted orderings of candidates. Hence, whichever state is realized, not all of the edges in the cycle can be winning edges. The worst-case for the auctioneer corresponds to a state where every edge in the cycle gets paid by \$ 1 except one, with $|C| - 1$ be the maximum payment to traders. Hence, if inequality (9.13) is satisfied, the auctioneer has non-negative worst-case profit by accepting the orders in the cycle. □

It can be shown that identifying such a non-negative worst-case profit cycle in an order graph G can be achieved in polynomial time.

Lemma 9.4.14. *It takes polynomial time to find a cycle in an order graph $G(V, E)$ that has the highest worst-case profit, that is*

$$\max_{C \in \mathcal{C}} \left(\sum_{e \in C} b_e - (|C| - 1) \right),$$

where $\mathcal{C}$ is the set of all cycles in G.

Proof. Because

$$\sum_{e \in C} b_e - (|C| - 1) = \sum_{e \in C} (b_e - 1) + 1 = 1 - \sum_{e \in C} (1 - b_e),$$

finding the cycle that gives the highest worst-case profit in the original order graph G is equivalent to finding the shortest cycle in a converted graph $H(V, E)$, where H is achieved by setting the weight for edge e in G to be $(1 - b_e)$.

Finding the shortest cycle in graph H can be done within polynomial time by resorting to the *shortest path* problem. For any vertex v in V, we consider every neighbor vertex w such that $(v, w) \in E$. We then find the shortest path from w to v, denoted as $path(w, v)$. The shortest cycle that passes vertex v is found by choosing the w such that $e_{(v,w)} + path(w, v)$ is minimized. Comparing the shortest cycle found for every vertex, we then can determine the shortest overall cycle for the graph H. Because the shortest path problem can be solved in polynomial time [26], we can find the solution to our problem in polynomial time. □

If the worst-case profit for the optimal cycle is non-negative, we know that there exists a match in G. However, the condition in lemma 9.4.13 is not a necessary condition for the existence of a match. Even if all single cycles in the order graph have negative worst-case profit, the auctioneer may accept multiple interweaving cycles to have positive worst-case profit. Figure 9.1 exhibits such a situation.

If the optimal indivisible match consists only of edge disjoint cycles, a natural greedy algorithm can find the cycle that gives the highest worst-case profit, remove its edges from the graph, and proceed until no more cycles exist. However, we show that such greedy algorithm can give a very poor approximation.

Lemma 9.4.15. *The greedy algorithm gives at most an $O(\sqrt{n})$-approximation to the maximum number of disjoint cycles.*

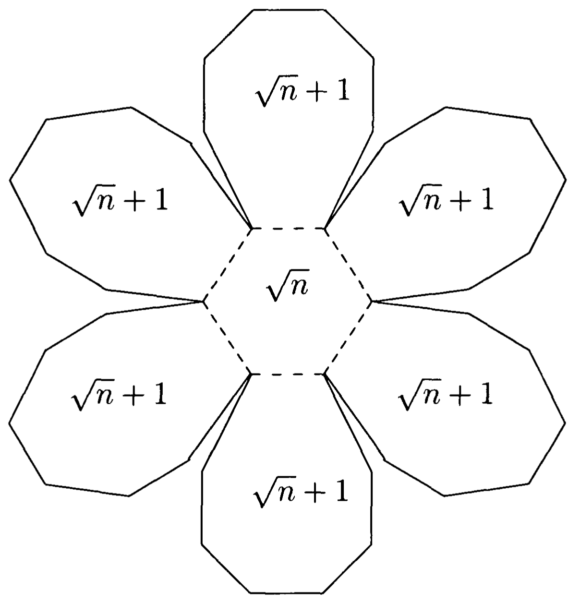


Figure 9.3. Graph with n vertices and $n + \sqrt{n}$ edges on which the greedy algorithm finds only two cycles, the dotted cycle in the center and the unique remaining cycle. The labels in the faces give the number of edges in the corresponding cycle.

Proof. Consider the graph in Figure 9.3 consisting of a cycle with $\sqrt{n}$ edges, each of which participates in another (otherwise disjoint) cycle with $\sqrt{n}+1$ edges. Suppose all edge weights are $(1-\epsilon)$. The maximum number of disjoint cycles is clearly $\sqrt{n}$, taking all cycles with length $\sqrt{n}+1$.

Because smaller cycles gives higher worst-case profit, the greedy algorithm would first select the cycle of length $\sqrt{n}$, after which there would be only one remaining cycle of length n. Thus the total number of cycles selected by greedy is 2 and the approximation factor in this case is $\sqrt{n}/2$. □

In light of Lemma 9.4.15, one may expect that greedy algorithms would give $\sqrt{n}$-approximations at best. Approximation algorithms for finding the maximum number of edge-disjoint cycles have been considered by Krivelevich, Nutov and Yuster [81, 131]. Indeed, for the case of directed graphs, the authors show that a greedy algorithm gives a $\sqrt{n}$-approximation [81]. When the optimal match does not consist of edge-disjoint cycles as in the example of Figure 9.3, greedy algorithm trying to finding optimal single cycles fails obviously.

9.5 Concluding remarks on markets with an auctioneer

At the beginning of this chapter we described some of the design challenges for auction-based prediction markets. In particular, one of the main design difficulties when the outcome space is large and betting on all individual outcomes is not feasible, is to propose a bidding language (which specifies what types of securities traders can bet on) that is expressive enough and yet leads to a tractable matching problem for the auctioneer (namely there is an efficient algorithm to find and match trading partners risklessly).

One of the surprising consequences of our results on the subset and pair-bidding languages in the context of permutation-type outcomes is that the expressiveness and tractability design goals are not necessarily conflicting. In particular, pair-bidding allowed traders to bet on ordered pairs of candidates, namely to bet on $\binom{n}{2} = O(n^2)$ securities and the corresponding auctioneer problem was NP-hard in all possible variants with divisible or indivisible orders. On the other hand, subset-bidding allowed traders to bet on which subset of players would come in a given position and vice versa, namely it allowed for a much richer set of (exponentially many) securities while at the same time admitting an efficient algorithm for solving the auctioneer problem.

Another notable feature is the interdisciplinary approach that we needed to use, to perform the economic analysis via a variety of classical computer science / combinatorial optimization problems.

One open direction for auction-based prediction markets would be to explore approximation algorithms for cases where the auctioneer matching problem is hard. Another necessary direction to make the designs practical would be to extend the offline to an online mechanism of submitting trade orders, in which the auctioneer has to decide upon receiving an order whether to accept it or not. This may not be possible to do in a risk-free manner, in which case it would be interesting to investigate the trade-off between risk (*e.g.,* quantified by expected or worst-case loss) and trading volume.

CHAPTER 10

Open questions: a market maker or an auctioneer?

The main purpose of prediction markets is to aggregate information and make a good prediction. An accurate forecast is valuable in many ways: it enables better business planning and decision-making, better policy analysis, etc. Since accurate information has tremendous financial value, it is reasonable for a prediction market to require a subsidy (the market maker loss) in order to facilitate trade and acquire information. An auctioneer on the other hand bears no financial risk; however, he may not be able to aggregate information as effectively. We summarize some of the pros and cons of both types of markets below.

Markets with a market maker. The main advantages of having a market maker are to:

+ Ensure liquidity in the market (traders willing to trade at the current price can always do so).
+ Make it easier to identify when trade is possible and to implement it. (There is no need to solve a complex optimization problem to find trading partners: everyone trades directly with the market maker at the posted price.)
+ Every security has a unique well-defined price at a given time, which directly translates into a probability estimate for the corresponding outcome.

The main disadvantage is the market maker loss, but as already discussed, this is not necessarily a flaw since it makes sense for the market maker to pay for accurate information aggregated in the market. Other challenges include the computation of the prices and probabilities according to the market maker's price-setting rule.

Markets with an auctioneer. The advantages of having an auction-based mechanism are:

+ The auctioneer by definition bears no risk and incurs no loss; in fact he may make a profit.
+ Traders set their own price in the orders (bids) they submit. Thus all traders have incentive to reveal their information or belief about the security values, not just traders who are willing to trade at one particular price as in the market maker setting.

The disadvantages of markets with an auctioneer include:

- The auctioneer needs to solve a complex optimization problem to find bilateral or multilateral trading partners.

- There may be no consistent interpretation of prices as probabilities. (For example, at a given time there may be multiple prices associated with the same security.)
- There is a trade-off between auctioneer risk and trading volume: there may be few or no orders that can be matched risklessly, and without trade there is no information flow.

In addition, a market with a market maker is sequential by definition. The auction settings we considered were offline, one-round auctions. Implementing a sequential design, in which the auctioneer has to accept or reject an order at the moment it is submitted presents a whole new challenge for how to formulate and solve the matching problem risklessly.

Common challenges and open questions. In the chapter with a market maker, we focused on computing best responses of the individual traders. It is still an open question to analyze the resulting equilibria, their existence and implementation. Similarly, in the chapter of prediction markets with an auctioneer, we focused on the auctioneer problem, taking as given the orders submitted by the traders. But how would the traders determine the optimal bids to place? It is open to analyze a game theoretic framework incorporating both the auctioneer and traders as players. Analyzing the possible equilibria is critical especially for understanding the incentives for long-term manipulation.

The automated market maker can be viewed as an online algorithm for accepting orders in the sequence they arrive. Could it be used to extend the auction-based prediction markets from the current off-line to an online setting? Is there a natural correspondence between the bidding language in our auction framework and a type of market maker, namely is there a correspondence between the design of the two types of prediction markets?

Bibliographic notes. The material in Chapter 8 is based on joint work with Rahul Sami [100] and Chapter 9 is based on joint work with Yiling Chen, Lance Fortnow and David Pennock [20].

Part III

Optimization in Economics: *Mechanisms and Algorithms for Network Routing*

CHAPTER 11

Motivation

In this part, we consider the interplay of optimization and economics in the context of network routing. We are motivated, in particular, by the network of autonomous systems in the Internet where each portion of the network is controlled by an Internet service provider, namely by a self-interested economic agent. The business incentives do not exist merely in addition to the computer protocols governing the network. Although they are not currently integrated in those protocols and are decided largely via private contracting and negotiations, these economic considerations are a principal factor that determines how packets are routed. And vice versa, the demand and flow of network traffic fundamentally affect provider contracts and prices.

The contributions of this part are the design and analysis of economic mechanisms for network routing. The mechanisms are based on first- and second-price auctions (the so-called Vickrey-Clarke-Groves, or VCG mechanisms).

We first analyze the equilibria and prices resulting from these mechanisms. In Chapter 13 we investigate the resulting prices from the celebrated Vickrey-Clarke-Groves truthful mechanism in which each edge on the cheapest path is paid the most it could have bid and still remained on the cheapest path. This mechanism is known to result in very high overpayments in which the packet ends up paying the selected route a price much higher than the cost of the next best alternative: we show that in more realistic network models this overpayment is insignificant. In Chapter 12 we consider instead first-price mechanisms in which edges are paid their bid, and show that the route payments under those mechanisms are much lower than the corresponding VCG payments.

Finally in Chapter 14 we investigate the compatibility of the better understood VCG-mechanisms with the current inter-domain routing protocols. We demonstrate the critical importance of correct modeling and analyze how it affects the complexity and algorithms necessary to implement the economic mechanisms.

CHAPTER 12

First-price auction mechanisms

In this chapter we consider variants on first price auctions: by these we mean any auctions in which the links on the winning paths are paid their bid amount. The designer still has considerable flexibility in designing the details of the auction mechanism.

We begin by exploring the sets of bids that are stable under a first-price auction mechanism. The most natural solution concept is that of a Nash equilibrium. We want to retain the property that agents can see each others' bids, so that the bidding could be performed through posted prices. Thus, mixed-strategy equilibria are not very meaningful for us. Unfortunately, we shall not necessarily have a Nash Equilibrium in pure strategies, as the following simple example shows. Consider a network of two parallel links, one of cost 2 and another of cost 1. Also, assume that ties are broken deterministically by assigning the flow to the link with cost 2. In this case, the lower-cost edge would bid less than 2; however, for any bid $2 - \epsilon$, it could always do better by increasing its bid by a further $\epsilon/2$. Hence, there is no pure Nash equilibrium in this case.

This motivates us to use the solution concept of ϵ-Nash equilibrium, in which no player can deviate in a way that improves his payoff by at least ϵ. Unfortunately, there is a drawback to this solution concept as well. In Figure 12.1, we see that the winning path may have a price higher than the cost of the best competitor. This defeats our goal of reducing customer overpayment. We might argue that this solution would not be sustained in practice, since the edges on the second lowest-cost path are likely to each reduce their price. This leads us to explore, in Section 12.2, the concept of a *strong* ϵ-Nash equilibrium, in which there is no *group* of agents who can deviate in a way that improves the payoff of each member by at least ϵ. We prove that a strong ϵ-Nash equilibrium always exists for any $\epsilon > 0$. We then prove an upper bound on the payment of any such equilibrium and show that the payment is essentially no more than that of the corresponding VCG payment, and often it is much less, as shown by Figure 13.1.

Although strong ϵ-Nash equilibria may solve some of the overpayment problem, we cannot guarantee that bidders will reach one. In particular, in the absence of knowledge about other bidders' costs, neither losing bidder in the example of Figure 12.1 may be willing to "blink first" and lower the price. Thus, in Section 12.3, we present a modified, randomized, first-price auction that explicitly drives the first-price auction towards a strong ϵ-Nash equilibrium.

Another drawback of first-price auctions is that, unlike the VCG mechanism, an edge's preferred bid may depend on the demand (e.g., if demand is high, an edge can bid higher and still hope to be needed). It is unreasonable to expect edges to delay setting prices until demands are made clear. Thus, in Section 12.4 we consider a model in which bidders set prices according to a *distribution* of possible demands. We show that, in this model, a simple first-price auction may not have an ϵ-Nash equilibrium. However, we design a first-price mechanism involving *two-parameter* bids that *does* have an ϵ-Nash equilibrium. We then sketch a mechanism that combines this two-parameter mechanism with the randomized mechanism of Section 12.3. For this combined mechanism, we can characterize the set of all ϵ-Nash equilibria, and thereby prove a bound on the total payment in any ϵ-Nash equilibrium.

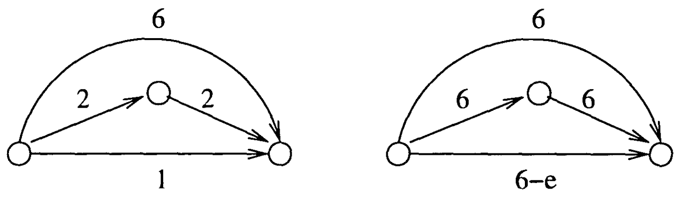


Figure 12.1. Costs (left) and Prices (right) in an ϵ-Nash equilibrium. The bottom edge wins and the price is higher than the cost of the second best path.

12.1 Definitions and notation

In the *path auction game*, there is a network G of strategic links, each with a privately-known true cost. All links have unit capacity. A customer wants to buy routes from a source S to a sink T in the network to guarantee that her integral amount of demand k can be routed. In order to do this, she defines a set of rules, or *mechanism*, that elicits bids from each agent and then allocates flow to each agent in a way that satisfies some natural incentive properties.

One plausible mechanism for this problem is the Vickrey-Clark-Groves (VCG) mechanism [127, 23, 59]. This mechanism is *truthful dominant strategy* or *strategyproof*, *i.e.*, the strategically best bid for an agent is his true cost, independent of the others' bids. Thus the bids solicited by the mechanism in an equilibrium are in fact the true costs of the agents. This enables the mechanism to allocate flow to the lowest *true cost* k-flow, a socially desirable goal in many settings. However, in order to guarantee that this allocation rule is strategyproof, the mechanism must pay a (possibly large) premium to all edges on the selected k-flow. One feature of dominant strategies is that all bargaining between the strategic agents (links, in our case) is eliminated; the overpayment to edges on the selected k-flow in the VCG mechanism can be thought of as a side-effect of this fact.

We analyze approaches to reducing the total payment by using a weaker solution concept of a pure strategy equilibrium, the *strong ϵ-Nash equilibrium* first introduced by Aumann [5] and used by Young [130].

Definition 12.1.1. *An* ϵ-Nash equilibrium *for a game is a set of strategies, one for each player, such that no player can unilaterally deviate in a way that improves her payoff by at least ϵ.*

A strong ϵ-Nash equilibrium *for a game is a set of strategies, one for each player, such that no group of players can deviate in a way that improves the payoff of each member by at least ϵ.*

In particular, we show that in our models, for any strong ϵ-Nash equilibrium set of bids, there is another strong ϵ-Nash equilibrium set of bids with the same allocation and payment scheme in which each agent bids within ϵ of his true cost unless he is allocated flow (in expectation), and he never bids below his true cost.

Our mechanism is a simple first-price auction. It elicits bids from each agent, computes the cheapest k-flow according to the bids, and then allocates the demand to that k-flow. We further assume that we have a deterministic tie-breaking rule, so that, if there is more than one cheapest k-flow, we take the lexicographically first integral one.

We consider two specific path auction games. In the *deterministic path auction game*, the user first announces k, his total demand. Then the edges announce bids and the user runs a first price auction to buy the necessary flow. It is easy to imagine that the assumptions of this model might be unrealistic in practice. Does a user really know his total demand at the time he begins the auction? In our second model, the *random path auction game*, the user announces a probability distribution on k. Then the edges announce bids. Finally, the user draws k according to this distribution and buys flow accordingly. In the rest of this section, we analyze upper and lower bounds on the payments in strong ϵ-Nash equilibria for each of these games.

Notation: For a graph G, let c be the vector of edge costs, let b be the vector of edge bids, and let $F_{\mathrm{w}}(k, G)$ be the set of edges in the minimum weight integral k-flow[1] in G with respect to edge weights w (if there is more than one minimum weight k-flow in G with respect to w, let $F_{\mathrm{w}}(k, G)$ denote the set of edges in the unique k-flow that wins the deterministic tie-breaking rule of the mechanism). We will refer to $F_{\mathrm{c}}(k, G)$ as the minimum cost k-flow and $F_{\mathrm{b}}(k, G)$ as the minimum price k-flow with respect to bid vector b. Finally, for any flow or edge set F, we define $W_{\mathrm{w}}(F)$ to be the weight of F with respect to edge weights w. We say $W_{\mathrm{c}}(F)$ is the cost of flow F and $W_{\mathrm{b}}(F)$ to be the price of flow F with respect to bid vector b. When the bids, costs, or graph are clear from the context, we shall sometimes drop them from the notation. As a shorthand, we sometimes write $C(F)$ instead of $W_{\mathrm{c}}(F)$, as well as $C(k)$ for the (cost of the) lowest cost k-flow. Finally, we denote the number of agents, or edges in G, by n.

12.2 Deterministic path auction game

Recall that in the *deterministic path auction game*, the user first announces k, his total demand. Then the edges announce bids and the user runs a first price auction to buy the necessary flow. We would like to analyze the payment properties of this mechanism. First, we prove that this mechanism has a strong ϵ-Nash equilibrium.

Theorem 12.2.1. *Any deterministic k-unit first price auction has a strong ϵ-Nash equilibrium.*

Proof. We construct a strong ϵ-Nash equilibrium as follows: Set the initial bid vector $\mathrm{b}^{\mathrm{i}} = \mathrm{c}$, *i.e.*, each edge bids its true cost initially. Order the edges in the graph in an arbitrary way. For each edge e in this order, if e is part of the current lowest price k-unit flow $F_{\mathrm{b}}(k, G)$, let e raise its bid until $W_{\mathrm{b}'}(F_{\mathrm{b}'}(k, G)) \geq W_{\mathrm{b}}(F_{\mathrm{b}}(k, G - \{e\})) - \epsilon/2$ (where $G - \{e\}$ denotes the graph G with edge e removed). Otherwise, let e's bid remain unchanged. Call the final bid vector b^{f}.

We claim b^{f} is a strong ϵ-Nash equilibrium for the deterministic k-unit first price auction. To show this, suppose the contrary, *i.e.*, there is a coalition S of edges in which each edge can improve its payoff by at least ϵ by changing its bid. Note that for any bid vector constructed during this

[1]The weight of this flow is equal to the weight of the minimum weight k-flow, *i.e.*, requiring integrality doesn't change the value of the optimal solution.

process, the auction always selects the same k-flow. Therefore, the edges which are not on the winning flow in $\mathbf{b}^{\mathbf{f}}$ are bidding their true cost and cannot bid lower. Furthermore, the edges which are on the winning flow will get a smaller payoff if they decrease their bid. Therefore no edge can benefit from lowering its bid. Thus, the edges in the coalition S can only raise their bids. Suppose the edges in $S \cap F_{\mathbf{b}^{\mathbf{f}}}$ increase their bids by a total of x units and the remaining edges in the coalition increase their bids by a total of y units (note $x, y > 0$). Call the new bid vector $\mathbf{b}$. In order for all edges in S to increase their payoff, $S \subseteq F_{\mathbf{b}}$. Thus $W_{\mathbf{b}}(F_{\mathbf{b}^{\mathbf{f}}}) = W_{\mathbf{b}^{\mathbf{f}}}(F_{\mathbf{b}^{\mathbf{f}}}) + x$ while $W_{\mathbf{b}}(F_{\mathbf{b}}) = W_{\mathbf{b}^{\mathbf{f}}}(F_{\mathbf{b}}) + x + y$. But then $W_{\mathbf{b}}(F_{\mathbf{b}}) > W_{\mathbf{b}}(F_{\mathbf{b}^{\mathbf{f}}})$ since $W_{\mathbf{b}^{\mathbf{f}}}(F_{\mathbf{b}^{\mathbf{f}}}) \leq W_{\mathbf{b}^{\mathbf{f}}}(F_{\mathbf{b}})$ by optimality of $F_{\mathbf{b}^{\mathbf{f}}}$. This contradicts the optimality of $F_{\mathbf{b}}$. □

Given the existence of strong ϵ-Nash equilibria, we can bound the payments in any such equilibrium. The main insight here (formalized in the next theorem) is that the *payment* per unit for sending k units of flow does not exceed the marginal *cost* of sending a $(k+1)$-st unit of flow. In order to develop some intuition for the proof, it is useful to first consider sending 1 unit of flow in a graph consisting of just two parallel edges from the source s to the sink t of costs a and b, $a > b + \epsilon$. The lower true cost edge must be allocated the flow in equilibrium since he can bid just under the true cost of the higher cost edge and be guaranteed a profit of at least ϵ. Therefore, by the conditions of a strong ϵ-Nash equilibrium, we can assume that the bid of the higher cost edge is at most ϵ more than his true cost and so the overpayment of any equilibrium will be at most $a + \epsilon - b$. The crux of this argument was to bound the bid of the winning path by the bid of an augmenting path. Since the augmenting path does not receive flow, we could show that without loss of generality the bid of this path should be close to its true cost. This proof idea easily extends to k-flows in general graphs, as can be seen below.

Theorem 12.2.2. *The total payment of the deterministic k-unit first price auction in a strong ϵ-Nash equilibrium is at most*

$$k\big[C(F_{\mathbf{c}}(k+1)) - C(F_{\mathbf{c}}(k))\big] + kn\epsilon,$$

where $\mathbf{c}$ *is the vector of true edge costs.*

Proof. Fix a strong ϵ-Nash equilibrium vector of bids $\mathbf{b}$ and define edge sets

$$\begin{aligned} E_+ &= \{e \in F_{\mathbf{c}}(k+1) - F_{\mathbf{b}}(k)\} \\ E_o &= \{e \in F_{\mathbf{c}}(k+1) \cap F_{\mathbf{b}}(k)\} \\ E_- &= \{e \in F_{\mathbf{b}}(k) - F_{\mathbf{c}}(k+1)\} \end{aligned}$$

E_+ is the subset of edges on an augmenting path that are not in the original flow $F_{\mathbf{b}}(k)$. We show that without loss of generality we may assume that these edges are bidding close to their true cost. To show this, consider a bid vector $\mathbf{b}'$ such that

$$b'_i = \begin{cases} \min\{b_i, c_i + \epsilon\} & \text{for } i \in E_+, \\ b_i & \text{for } i \notin E_+. \end{cases}$$

We want to argue that $W_{\mathbf{b}'}(F_{\mathbf{b}'}(k)) = W_{\mathbf{b}}(F_{\mathbf{b}}(k))$. First we show $F_{\mathbf{b}}(k) = F_{\mathbf{b}'}(k)$. Suppose not. Let $E'_+ = E_+ \cap F_{\mathbf{b}'}(k)$ be the set of edges in the new lowest price flow that are also in E_+. We have only changed the bids of the edges in E_+, so if E'_+ is empty then $F_{\mathbf{b}}(k) = F_{\mathbf{b}'}(k)$ (this assumes some consistency properties of the tie-breaking rule). If E'_+ is nonempty, then we can consider a bid vector $\mathbf{b}''$ constructed from $\mathbf{b}$ in which we only decrease bids of edges in E'_+:

$$b''_i = \begin{cases} \min\{b_i, c_i + \epsilon\} & \text{for } i \in E'_+, \\ b_i & \text{for } i \notin E'_+. \end{cases}$$

Since by our assumption the winning flow has changed, we must have $b''_i = c_i + \epsilon < b_i$ for a nonempty subset E''_+ of E'_+. Under this new bid vector, $W_{\mathbf{b}''}(F) \geq W_{\mathbf{b}'}(F)$ for any flow F since $\mathrm{b}'_\mathrm{i} \leq \mathrm{b}''_\mathrm{i}$ for all edges i. By construction, $W_{\mathbf{b}''}(F_{\mathbf{b}'}(k)) = W_{\mathbf{b}'}(F_{\mathbf{b}'}(k))$ and so, by the consistency of the tie-breaking rule, $F_{\mathbf{b}'}(k) = F_{\mathbf{b}''}(k)$. Thus, under the bid vector $\mathbf{b}$, the set of edges E''_+ can form a coalition in which each member bids ϵ above its true cost and all members profit by ϵ. This contradicts the fact the $\mathbf{b}$ was a strong ϵ-Nash equilibrium.

Now, noting that $W_{\mathbf{b}'}(F_{\mathbf{b}'}(k)) = W_{\mathbf{b}'}(F_{\mathbf{b}}(k)) = W_{\mathbf{b}}(F_{\mathbf{b}}(k))$, it suffices to bound $W_{\mathbf{b}'}(F_{\mathbf{b}'}(k))$. Consider the (non-integral) flow $(k/(k+1))F_{\mathbf{c}}(k+1)$, *i.e.,* the flow which sends $k/(k+1)$ units of flow along the flow paths determined by $F_{\mathbf{c}}(k+1)$. Since $F_{\mathbf{b}'}(k)$ is a lowest price k-flow,

$$\left(\frac{k}{k+1}\right) W_{\mathbf{b}'}(F_{\mathbf{c}}(k+1)) - W_{\mathbf{b}'}(F_{\mathbf{b}'}(k)) \geq 0.$$

This reduces to

$$\left(\frac{k}{k+1}\right) W_{\mathbf{b}'}(E_+) - \left(\frac{1}{k+1}\right) W_{\mathbf{b}'}(E_o) - W_{\mathbf{b}'}(E_-) \geq 0$$

which, solving for $W_{\mathbf{b}'}(E_o) + W_{\mathbf{b}'}(E_-)$, gives

$$\begin{aligned} W_{\mathbf{b}'}(F_{\mathbf{b}'}(k)) &= W_{\mathbf{b}'}(E_o) + W_{\mathbf{b}'}(E_-) && (12.1)\\ &\leq k(W_{\mathbf{b}'}(E_+) - W_{\mathbf{b}'}(E_-)) && (12.2)\\ &\leq k(W_{\mathbf{c}}(E_+) + n\epsilon - W_{\mathbf{c}}(E_-)) && (12.3)\\ &\leq k(W_{\mathbf{c}}(F_{\mathbf{c}}(k+1)) - W_{\mathbf{c}}(F_{\mathbf{b}'}(k)) + n\epsilon) && (12.4)\\ &\leq k(W_{\mathbf{c}}(F_{\mathbf{c}}(k+1)) - W_{\mathbf{c}}(F_{\mathbf{c}}(k)) + n\epsilon) && (12.5) \end{aligned}$$

where 12.3 follows from the fact that for any edge $b'_i \geq c_i$ and for all $i \in E_+$, $b'_i \leq c_i + \epsilon$; and 12.5 follows from the optimality of $F_{\mathbf{c}}(k)$ with respect to $\mathbf{c}$. □

In addition, it is easy to see that this bound is tight. Consider a graph with $(k+1)$ parallel edges where the cost of the bottom k edges is c and the cost of the remaining top edge is $c' > c$. Let all k lower cost edges bid $c' - \epsilon$ for a small $\epsilon > 0$, so their bid is less than the bid of the remaining higher cost edge (whose bid is at least c'). The minimum price k-flow with respect to this bid vector will use the bottom k edges for a total price of $k(c' - \epsilon)$ which approaches $k(C(F_{\mathbf{c}}(k+1)) - C(F_{\mathbf{c}}(k)))$.

Finally, we emphasize two properties of our mechanism. The first property states that the total payment of our first price mechanism in a strong ϵ-Nash equilibrium is at most $kn\epsilon$ more than the VCG payment for the same graph in a Nash equilibrium. The second property states that the social welfare of the resulting solution is an additive approximation to the optimum social welfare.

Theorem 12.2.3. *Given a graph G with source S and sink T, the VCG payment for k units of flow from S to T is at least $k(C(F_{\mathsf{c}}(k+1)) - C(F_{\mathsf{c}}(k)))$.*

Theorem 12.2.4. *In a strong ϵ-Nash equilibrium* $\mathbf{b}$, $C(F_{\mathbf{b}}(k)) \leq C(F_{\mathsf{c}}(k)) + \epsilon n$ *(i.e., the strong ϵ-Nash equilibria of the first price auction are approximately efficient).*

12.3 Implementation in ϵ-Nash

The simple first-price auction may have costly ϵ-Nash equilibria, as shown in the example in Figure 12.1. In Section 12.2 we used the ϵ-strong Nash solution concept to get around this problem. However, assuming that the bidders will reach an ϵ-strong Nash equilibrium is perhaps too strong an assumption: it requires extensive coordination between agents. In this section, we present a variant of the mechanism in which every ϵ-Nash equilibrium results in a low price.

One idea to achieve this is to pay a bonus to edges that increases as their bid decreases. This encourages edges to submit low bids. However, this has the side-effect of incentivizing edges to bid even below their true cost, as long as they remain off the winning path. This would make the bargaining problem that links must solve much more complex, as it would include bargains between off-path and on-path links. Alternatively, we could instead send flow on each edge with some probability that increases as the bid decreases. Thus an edge will not bid below its true cost, but it might be incentivized to bid very high. Using a combination of these two ideas, we can construct a payoff function such that an edge will bid close to its true cost if it is not on the lowest true cost flow. If the bonuses and probabilities are small enough, then these bonus payments will not be very large, and we can prove a bound on the total payment of the mechanism similar to that in Theorem 12.2.2.

We achieve this result by making the mechanism outcome a *lottery* over paths instead of a single path: every edge is on a selected path with at least a small probability, and edges off the shortest path are given an incentive to bid their true cost. This is known as *virtual implementation* in the economics literature (see, *eg.* Jackson [70]). We assume that there is a value B such that no edge bids more than B. (Alternatively, B can be the maximum amount that the buyer is willing to pay.) Further, we assume that the edges are risk-neutral. The mechanism is given in Figure 12.2. The mechanism starts by computing a collection of paths $\{P_e\}$. We discuss the computation of this collection in Section 12.3.1. The mechanism then invites a bid b_e from each edge e. The lowest-price path is almost always picked; however, with a small probability, one of the paths from the collection is picked instead. In addition, each edge is paid a small bonus that depends on the bids. The selection probability and bonus are chosen to ensure that the optimal strategy for every edge, which is *not* on the lowest-price path, is to bid its true cost. For simplicity, we present the mechanism and analysis for a single unit flow; the results can easily be extended to any constant $k > 1$ units of flow. First we note that ϵ-Nash equilibria exist in this mechanism; indeed the same construction as in Theorem 12.2.1 yields an ϵ-Nash equilibrium.

Lemma 12.3.1. *For any cost vector* $\mathbf{c}$ *and any $\epsilon > 0$, an ϵ-Nash equilibrium always exists.*

Given the existence of ϵ-Nash equilibria and total payoff function to each edge (sum of bonus and expected selection payoff), we can bound the bid of the edges not on the lowest true-cost

1. For each edge e, find P_e, a path from s to t through e. Let $\mathcal{P} = \{P_e\}_{e \in G}$. Note that an edge e may appear in multiple paths in $\mathcal{P}$.
2. Invite bids $\mathbf{b} = (b_1, \ldots, b_e, \ldots, b_n)$ from the edges.
3. For each path $P \in \mathcal{P}$, compute
$$\sigma_P = \alpha - \tau \sum_{e \in P} b_e$$
4. Select each path $P \in \mathcal{P}$ with probability σ_P; with probability $(1 - \sum_{P \in \mathcal{P}} \sigma_P)$, select the lowest price path. Call the selected path P^*. Pay each edge $e \in P^*$ its bid b_e.
5. Pay each edge $e \in G$ the sum $f_e(\mathbf{b}) = \sum_{P \in \mathcal{P}, P \ni e} f_e^P(\mathbf{b})$, where
$$f_e^P(\mathbf{b}) = \alpha(B - b_e) + \tau b_e \sum_{j \in P} b_j - \tau \frac{b_e^2}{2}$$
(This payment is in addition to any payment edge e may get in step 4.)

Figure 12.2. Mechanism FP2. The parameters α and τ are selected to be small positive constants such that $\alpha < n^{-2}B^{-1}$ and $\tau < \alpha n^{-1} B^{-1}$.

path by examining their optimal bid. Note that the bonus increases as the bid decreases, but the expected selection payment decreases as the bid decreases. Intuitively, we design the bonus and selection probabilities so that the total payoff function is maximized when $b_i = c_i$. Note that if an edge is selected, it incurs its true cost. In this way, the true cost automatically enters his expected payoff function—the mechanism does not need to know the cost in order to achieve the maximum at $b_i = c_i$.

By evaluating the expected payoff of an off-path link in mechanism FP2, we can show:

Lemma 12.3.2. *Let* $\mathbf{b}$ *be an* ϵ*-Nash equilibrium bid vector in the mechanism FP2. Then, for any edge* e not *on the lowest-price path with bids* $\mathbf{b}$, $b_e \in [c_e - \sqrt{2\epsilon/\tau}, c_e + \sqrt{2\epsilon/\tau}]$.

Now, we observe that the values α and τ can be chosen small enough to make the probabilities $\{\sigma_P\}$ and bonuses $f_e^P(\mathbf{b})$ arbitrarily small. Thus, the total payment to edges not on the shortest path is very small. The bound on the payment of mechanism FP2 is more sensitive to the value of ϵ because edges not on the lowest-price path get very small payments in expectation. However, we can show that, in the limit as $\epsilon \rightarrow 0$, the maximum expected payment in any Nash equilibrium is bounded by the same constant as before.

Observing that as $\epsilon \rightarrow 0$, $\sqrt{2\epsilon/\tau} \rightarrow 0$, we get the following result:

Theorem 12.3.3. *Choose any* $\alpha < n^{-2}B^{-1}, \tau < \alpha n^{-1}B^{-1}$. *For these values of* α *and* τ,
$$\lim_{\epsilon \rightarrow 0} \max_{\epsilon\text{-}NE\ \mathbf{b}} \{\textit{Total payments with bids } \mathbf{b}\} \rightarrow C(2) - C(1) + 3\alpha n^2 B.$$

12.3.1 Computing the set of covering flows $\{P_e\}$

Recall that the mechanism FP2 needs to compute a set of paths $\{P_e\}$, where P_e is a path from s to t that uses edge e. If e is to be relevant to the path auction, such a path must exist; however, it is not always straightforward to compute. In particular, if the network is a general *directed* graph, it is NP-hard to compute such a path since it reduces to the two disjoint paths problem, which is NP-complete [49].

However, there are many interesting classes of graphs for which it is possible to compute such a path P_e in polynomial time, including undirected graphs and directed acyclic or planar graphs [49]. We can also modify the mechanism to ask each bidder to exhibit such a path, thus transferring the computational burden on to the bidders. Also, these paths may be precomputed and used in many executions of the mechanism—they do not depend on the costs or bids.

Another possibility is to use a set of covering paths that do not all terminate at t—this can be easily computed, even for general directed graphs. Then, if the path is picked, some arbitrary traffic is sent along this path. After this "audit" traffic has been delivered, the lowest-price path is used for the intended traffic from s to t. As long as the mechanism can verify that the traffic is correctly delivered, the edges would still have an incentive to bid as specified. Similarly, if we could verify the exact path that the traffic used, we could use non-simple paths to cover the edges; again, a set of non-simple covering paths can easily be found.

12.4 Distribution on demands

In the previous sections, we studied first-price auctions to meet a known demand, we argued that they had stable Nash equilibria, and showed how to adjust this mechanism so that the equilibria chosen by the user had relatively small overpayments. However, in practice, it may not be possible to defer the setting of prices until the demand is known. In this section, we examine the problem of achieving stable prices without advance knowledge of the demand. In particular, we assume that the edges know only of some *probability distribution* over the possible demands. Ideally, we would like our results for first-price auctions with known demand to carry over. For example, we proved in Section 12.2 that a first price auction for k units of demand led to a payment of $\Pi_k = k[C(F_c(k+1)) - C(F_c(k))]$. It is thus natural to hope that the same mechanism operating over *random* k is also stable, with expected payment $E_k[\Pi_k]$. This turns out to be false—in fact, we show in Section 12.4.1 that the simple first-price auction mechanism described previously has *no* ϵ-Nash equilibria. Intuitively, this is because edges must trade off the *probability* of receiving flow with the *profit* of receiving flow. With a high bid, the profit is large, but the probability of winning the auction is low. If the other bids are also high, an edge will prefer to lower its bid to win with a higher probability. This will lead other edges to lower their bids so as to restore their high winning probability. Now, however, the first edge will increase its bid so as to increase its profit at the expense of its winning probability, and so a cycle emerges in the bidding strategies. So we need to turn to more complex mechanisms.

We exhibit a mechanism involving *two-parameter bids* that, unlike the single-parameter first-price mechanism, *does* have ϵ-Nash equilibria. Intuitively, a two-parameter mechanism gets around the problem of a single-parameter mechanism by letting the edges express their preferences over the entire price-probability space. The mechanism allows an edge to bid a "price" that depends on its winning probability; this prevents the bidding cycles that occur with single-parameter bids. Furthermore, using an indifference-breaking technique similar to that of Section 12.3, we are able to restrict the set of equilibria to ones with bounded user payments. The bound is not quite the $E_k[\Pi_k]$ we hoped to achieve, but does bear a clear resemblance to it.

12.4.1 No equilibrium with 1-parameter bidding

In this section, we analyze the scenario in which the demand is random with a known distribution and the bidders (links) have to commit to a price before the demand is revealed, and there is deterministic tie-breaking. Subsequently, the demand is revealed to be k, and the k lowest-priced paths are picked. We show that there is in general no ϵ-Nash equilibrium in pure strategies for this game.

Consider a graph with four parallel links W, X, Y, and Z between the source and the sink, with true costs w, x, y, and z respectively. The demand is either 1, 2 or 3; for simplicity, let the probability of each demand value be $\frac{1}{3}$. Assign the costs such that $w+50\epsilon < x+42\epsilon = y+12\epsilon = z$.

Theorem 12.4.1. *There is no pure-strategy ϵ-Nash equilibrium for this game.*

The proof repeatedly uses the ϵ-Nash conditions to show that, at any bid vector, one of the following must hold: (1) There is an agent who would gain by raising its bid; or (2) There is an agent who would gain by undercutting another agent to win with a higher probability.

12.4.2 Equilibrium with 2-parameter bidding

In section 12.4.1, we saw that when the demand is a random variable with a known distribution, a simple first-price auction may not have an ϵ-Nash equilibrium. In this section, we present a different auction model, in which agents' bids are pairs of values, and show that it has a nonempty set of ϵ-Nash equilibria.

To prove that the bidding game induced by this auction has a strong ϵ-Nash equilibrium, we construct a cooperative game model of the auction. We show that the cooperative game has a nonempty *core*. We then connect the cooperative game to the actual bidding game, and show that the path auction has an ϵ-Nash equilibrium corresponding to any core element.

The model is as follows: The demand can take any integral value in the range $[1, r]$, where r is a positive integer. Further, there is a known prior distribution on the demand values; say that the demand is k with probability p_k, for $k = 1, 2 \ldots, r$. We assume for simplicity that $p_k > 0$ for all k; our results easily extend to a situation in which $p_k = 0$ for some values of $k \in \{1, \ldots, r\}$. The agents' bids are pairs of numbers: each agent i bids a pair $\tilde{a} = (\tilde{c}_i, \tilde{u}_i)$, where $\tilde{c}_i$ is interpreted as i's reported cost, and $\tilde{u}_i$ is interpreted as i's demanded profit.

The mechanism receives the bids and announces flows $F_1, F_2, \ldots, F_r$ for each possible demand value. We call the collection $\mathcal{F} = \{F_1, F_2 \ldots, F_r\}$ a candidate solution or simply a flow. We also identify a flow $\mathcal{F}$ with the set of links in the union $F_1 \cup F_2 \cup \cdots \cup F_r$, and say that $i \in \mathcal{F}$ if $i \in F_k$ for some k.

For each $i \in \mathcal{F}$, the mechanism calculates the probability that i is in the winning flow, $\rho_i = \sum_{\{k | i \in F_k\}} p_k$. Later, the actual demand transpires: suppose that the demand turns out to be k. The mechanism uses the links in F_k to route the flow, and pays each link $i \in F_k$ a sum of $\tilde{c}_i + \frac{\tilde{u}_i}{\rho_i}$. Consider any link i selected in some flow. If $\tilde{c}_i = c_i$ (*i.e.,* if i bid its true cost), her expected profit would be $\tilde{u}_i$. Given the input bid pairs, the mechanism selects a set of flows $F_1, F_2, \ldots, F_r$ that minimizes the total expected payments. This can be expressed in terms of solving an integer program.

As before, we use $W_c(\mathcal{F})$ or $C(\mathcal{F})$ for short, to denote the total expected cost of a solution $\mathcal{F} = (F_1, \ldots, F_r)$ when the individual link costs are c, and $\text{W}_{\tilde{a}}(\mathcal{F})$ to denote the price of the flow $\mathcal{F}$ when the bids are $\tilde{a}$. We denote the mechanism output (*i.e.,* the min-price flow) by $\hat{\mathcal{F}}(\tilde{a})$.

12.4.3 The cooperative game $\mathcal{G}$

In this section, we define a cooperative game based on any specific instance of our 2-parameter mechanism, and prove that it has a nonempty core. This cooperative game is introduced only for strategic analysis of the mechanism. It is not explicitly played by the agents, but helps to shed light on the agents' strategies in the two-parameter auction.

Definition 12.4.2. *Given a set of players P, a* cooperative game *is defined by a* characteristic function $v : 2^P \to \Re_{\geq 0}$, *with $v(\emptyset) = 0$. For any $S \subseteq P$, $v(S)$ is called the* value *of the set S.*

Given a directed graph G with distinguished source and sink, and a true cost c_i for each link $i \in G$, we define the cooperative game $\mathcal{G}$ as follows:

The set of players in the game $\mathcal{G}$ is $P = \{0, 1, \cdots, n\}$, where each $i > 0$ is the player corresponding to link i, and 0 is a special player corresponding to the customer. Let Z be the customer's budget, and assume that Z is large enough to be irrelevant; $Z > r \times$ *cost of minimum-cost* $(r+1)$-*flow* is sufficient. For each set $S \subseteq P, S \neq \emptyset$, define the value $v(S)$ of S in $\mathcal{G}$ as follows:

If S does not contain a r-unit flow from s to t, $v(S) = 0$. If S contains the customer 0, as well as all edges on a k-unit flow from the source to the sink, $v(S)$ is defined to be the optimal value of the linear program given below:

Define $\delta_{i,S}$ to be the indicator of i in S, *i.e.,* $\delta_{i,S} = 0$ if $i \notin S$ and $\delta_{i,S} = 1$ if $i \in S$. Also, for any node α in the network, we use the notation $\text{In}(\alpha)$ to denote the set of incoming edges, and $\text{Out}(\alpha)$ to denote the set of outgoing edges. Then,

$$\begin{array}{ll}
v(S) = \max\left\{Z - \sum_{k=1}^{r}\left[p_k \sum_{i>0} c_i x_{ik}\right]\right\} & \\
\text{Subject to:} & \\
\sum_{i\in\text{Out}(\alpha)} x_{ik} - \sum_{i\in\text{In}(\alpha)} x_{ik} = 0 & \forall k \forall \alpha \neq s, t \\
\sum_{i\in\text{Out}(s)} x_{ik} - \sum_{i\in\text{In}(s)} x_{ik} - k = 0 & \forall k \\
x_{ik} \leq \delta_{i,S} & \forall k, \forall i \geq 0 \\
x_{ik} \geq 0 & \forall k, \forall i \geq 0
\end{array} \tag{12.6}$$

This linear program is interpreted as follows: For any link i, and any demand value k, the variable x_{ik} indicates the flow along i in F_k. Intuitively, the value of a set S is related to the net surplus that is created when only the agents in set S are involved in the flow. If S does not contain the customer and a r-unit flow, $v(S)$ is defined to be 0. Thus, only sets that contain at least one candidate solution are assigned a positive value.

We also note that if $S = P$, then the linear program has an integral optimal solution corresponding to an integral min-cost k-flow for each k. In other words, there is a solution in which x_{ik} is either 0 or 1 for all i and k. It is also clear that $v(S) \leq v(P)$ for all $S \subseteq P$.

Thus, the function $v(S)$ defines a finite, nonnegative value for each coalition set S and hence it is the characteristic function of a valid cooperative game $\mathcal{G}$.

Our analysis is centered on the concept of the *core* of a cooperative game. Loosely speaking, the core of a cooperative game consists of all ways to divide up the overall value $v(P)$ among the agents such that no group S has reason to be unhappy—*i.e.,* S attains a combined utility of at least $v(S)$. Formally, the core is defined as follows:

Definition 12.4.3. *A vector $u = (u_0, u_1, \ldots, u_n)$ is in the* core *of the game $\mathcal{G}$ if and only if it satisfies all of the following:*

$$\begin{array}{rcl} \forall i & u_i \geq 0 & \textit{and} \\ & \sum_{i \in P} u_i = v(P) & \textit{and} \\ \forall S \subseteq P & \sum_{i \in S} u_i \geq v(S). & \end{array}$$

In general, the core of a cooperative game might be the empty set. However, we can prove that this is not the case for the game $\mathcal{G}$:

Lemma 12.4.4. *The game $\mathcal{G}$ has a nonempty core.*

Proof. Consider any division of $v(P)$ among the players. We show that there is at least one such division that satisfies all the core constraints. For any set S with $v(S) = 0$, the core constraint is trivially satisfied. Now, consider a set S with $v(S) > 0$. The linear program defining $v(S)$ can be summarized in the form $\max\{x \cdot l\}$ subject to $xH = 0, xA \leq b^S$, and $x \geq 0$, where x is a vector of all the variables, H and A are matrices independent of S, and b^S is a 0-1 vector representing the capacity constraints for set S. Then, the dual of the linear program (12.6) is the following linear program:

$$\begin{array}{rl} & v(S) = \min\{b^S \cdot y\} \\ \text{Subject to:} & \\ & Ay + Hz \geq l \\ & y \geq 0 \end{array} \tag{12.7}$$

Now, consider the dual program that defines $v(P)$, *i.e.,* the value of the set containing the customer and all the links. Let $(\hat{y}, \hat{z})$ denote an optimal solution to this problem. Now, define $u_i = b^{\{i\}} \cdot \hat{y}$ for all i. Recall that b^S is a 0-1 vector, with 1s in precisely those equations that involve some $i \in S$; thus, $b^S = \sum_{i \in S} b^{\{i\}}$. Then, as $\hat{y} \geq 0$, we have $u_i \geq 0$, and

$$\sum_{i \in P} u_i = b^P \cdot \hat{y} = v(P).$$

Next, observe that for any set $S \subseteq P$, the solution $(\hat{y}, \hat{z})$ is also feasible in the dual of the program (12.7) defining $v(S)$. Thus, we have

$$\sum_{i \in S} u_i = b^S \cdot \hat{y} \geq v(S).$$

Thus, the vector u is in the core of the game $\mathcal{G}$. □

12.4.4 Existence of an ϵ-Nash equilibrium

We now show that given any point u in the core of this game, we can perturb it slightly to get a vector of bid pairs $\tilde{a}$ that is an ϵ-Nash equilibrium of the bidding game. We use the game $\mathcal{G}$ to draw conclusions about the bidding game induced by the mechanism.

Theorem 12.4.5. *Let u be any vector in the core of $\mathcal{G}$ that minimizes the value of u_0. Then, for any $\epsilon > 0$, the bid profile defined by*

$$a_i^- = (c_i, u_i^- = \max\{0, u_i - \frac{\epsilon}{2n}\})$$

for each link i is an ϵ-Nash equilibrium.

Proof. Suppose a^- is not an ϵ-Nash equilibrium. Then, there is some i such that i can change her bid to increase her payoff by ϵ. Let (c', u') be i's successful strategy, and let a' denote the bid profile given by $a_i' = (c', u')$ and $a_j' = a_j^-$ for all $j \neq i$. Let $\mathcal{F}' = \hat{\mathcal{F}}(a')$; it must be the case that $i \in \mathcal{F}'$.

We can show that there is a near-optimal flow $\mathcal{F}''$ such that $\mathcal{F}''$ does not use i. More specifically, $\mathrm{W}_{a^-}(\mathcal{F}'') \leq \mathrm{W}_{a^-}(\mathcal{F}') + \epsilon/2$. As $i \notin \mathcal{F}''$, we have $\mathrm{W}_{a'}(\mathcal{F}'') = \mathrm{W}_{a^-}(\mathcal{F}'')$. However, $i \in \mathcal{F}'$, and so $\mathrm{W}_{a'}(\mathcal{F}') \geq \mathrm{W}_{a^-}(\mathcal{F}') + \epsilon$. Thus, we get $\mathrm{W}_{a'}(\mathcal{F}') > \mathrm{W}_{a'}(\mathcal{F}'')$, which contradicts the assumption that $\mathcal{F}' = \hat{\mathcal{F}}(a')$. □

We are working on strengthening Theorem 12.4.5 to show that this bid profile is indeed a *strong* ϵ-Nash equilibrium. This seems plausible given the results of Young [130]; however, the strategy space in our 2-parameter game is richer than the strategy space in Young [130].

12.4.5 Randomized 2-parameter Auction

The mechanism presented in Section 12.4.2 has an ϵ-Nash equilibrium corresponding to every core allocation, but we cannot guarantee that there are no other ϵ-Nash equilibria. As a result, it was not possible to bound the total payoff to the edges. In this section, we consider a slightly modified mechanism in which we add a small random payment, as in Section 12.3. We prove that, with this modification, it is possible to bound the total payment.

The *Randomized 2-parameter Auction* is constructed as follows. As earlier, the edges' bids are pairs $\tilde{a}_i = (\tilde{c}_i, \tilde{u}_i)$. The mechanism has two components:

1. **The 2-parameter mechanism.** This mechanism is conducted exactly as described in Section 12.4.2 with parameters α, τ, and B set as before.

2. **The randomized audit.** For edges on a random source-destination path, the payoff is based entirely on the $\tilde{c}_i$ component of the bid, and is constructed as in Section 12.3. The parameters α, τ, and B are as defined in Section 12.3. To simplify the analysis, we assume that the randomized component results in a payoff function of the following form: If an edge has true cost c_i and bids $(\tilde{c}_i, \tilde{u}_i)$, its expected payoff from this component is $g(\tilde{c}_i) = \tau[c_i\tilde{c}_i - \frac{\tilde{c}_i^2}{2}]$. The exact form of the payoff was derived in the proof of Lemma 12.3.2, and has the same shape; the key aspect for us is that this function is maximized at $\tilde{c}_i = c_i$.

We also need to ensure that, for all edges i not in the winning solution, $\tilde{u}_i$ is 0 (or close to zero). We assume that the mechanism simply rejects bid profiles that do not meet this condition. Alternatively, we could impose a small tax on the $\tilde{u}_i$ component of the bid. We can now prove a useful lemma, which shows that *all* edges are nearly truthful about their costs in equilibrium:

Lemma 12.4.6. *Let $\tilde{a} = (\tilde{c}, \tilde{u})$ be an ϵ-Nash equilibrium of the Randomized 2-parameter Auction. Then, for all i,*

$$c_i - \sqrt{2\epsilon/\tau} \leq \tilde{c}_i \leq c_i + \sqrt{2\epsilon/\tau}.$$

Using the fact that the costs are nearly truthful, we can show that their utility values are nearly in the core, and hence, derive the following bound on the total payment.

Theorem 12.4.7. *Let $\tilde{a} = (\tilde{c}, \tilde{u})$ be any ϵ-Nash equilibrium of the Randomized 2-parameter Auction. Let $\mathcal{F}$ be a lowest-cost flow, and let F_{r+1} be a lowest-cost $(r+1)$-flow. Then, the total price paid by the customer in the randomized 2-parameter auction is at most*

$$\left[\sum_{j=1}^{r} j p_j C(F_{r+1})\right] - rC(\mathcal{F}) + nr\sqrt{2\epsilon/\tau} + 3\alpha n^2 B.$$

The result of Theorem 12.4.7 stands in an interesting relation to that of Theorem 12.2.2. We do not achieve the intuitively appealing upper bound given by the expectation of the bounds on the deterministic auction in Section 12.2:

$$E_j[\Pi_j] = \sum_{j=1}^{r} j p_j (C(F_{j+1}) - C(F_j)).$$

(Proving this stronger bound is an interesting problem for future work.) Instead, we achieve:

$$\sum_{j=1}^{r} r p_j (C(F_{r+1})(j/r) - C(F_j)).$$

In other words, the external multiplier j is replaced by r (a larger quantity), while in the first term the quantity $C(F_{j+1})$ is replaced by $C(F_{r+1})(j/r)$, which can also be larger because the cost of j units of flow is a convex function of j. Our Theorem 12.4.7 is therefore weaker than Theorem 12.2.2, but it does have a similar overall structure.

CHAPTER 13

Second-price (VCG) mechanism

The first-price auctions from Chapter 12 open the possibility of paying less than VCG auctions, but they do so by sacrificing valuable properties of the VCG mechanism. In particular, in a first-price auction, a risk-neutral edge may have incentive to lie, bidding a price higher than its cost. Also, in the absence of a dominant strategy, it may be necessary for bidders to communicate and bargain to achieve a stable set of bids, similarly to the current system of complex private contracting and negotiation.

In this section, we show that the VCG payments are not so high for networks of practical interest and thus the mechanism would also be attractive from the users' point of view. We first prove that the VCG Mechanism has constant expected overpayment for the Erdős-Renyi random graphs with unit costs and we demonstrate simulation results which suggest constant overpayment for power-law graphs and, in contrast, high overpayment for the much more connected complete graphs.

The VCG mechanism pays each edge on a winning path an amount equal to the highest bid with which the edge could still have won, all other bids being unchanged. The mechanism has the attractive property that each link's dominant strategy is to bid exactly its cost. Thus, no bargaining or communication between bidders is required to stabilize on bids. Also, the consumer does end up using the path of lowest true cost, which can be seen as optimizing social utility or the available system resources.

On the negative side, the VCG mechanism can lead to the customer paying far more than the true cost of the cheapest path. The tendency to overpay is increased in path auctions (as compared to simple auctions) because a bonus needs to be paid to every agent on the winning path. Thus, the payment to the lowest-cost path may even greatly exceed the cost of the second-cheapest path. For example, in Figure 13.1, VCG selects the bottom path and pays 4 to it, even though the alternate path has cost 3.

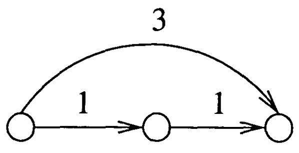


Figure 13.1. Any ϵ-Nash equilibrium selects the lower path and pays $3 - \epsilon$ while VCG pays 4 for it.

13.1 Definitions and the VCG mechanism

For a given source S and destination T, denote by $LCP(G,S,T)$ the cost of the lowest cost path from S to T in the graph G. The following theorem from Feigenbaum *et al.* [39] serves as our formal definition of the VCG Mechanism on networks.

Theorem 13.1.1. *[39][VCG Mechanism] When routing picks lowest-cost paths and the network is biconnected, there is a unique strategyproof pricing mechanism that gives no payment to edges that carry no transit traffic. The payment to each edge e on the lowest cost path from S to T is given by*

$$v(e,S,T) = cost(e) + LCP(G-e,S,T) - LCP(G,S,T). \tag{13.1}$$

$(G-e)$ above stands for the graph G with edge e deleted. Note that each edge on a winning path is paid its cost plus an extra term $LCP(G-e,S,T) - LCP(G,S,T)$, which we shall refer to as *bonus*. The bonus payment to an edge is precisely the marginal cost reduction of a route from adding the edge to the network. Another interpretation by Elkind [34] is that $v(e,S,T)$ is the threshold bid of an edge e, *i.e.*, the highest possible bid that the edge may announce and still be on the winning lowest-cost path from S to T.

Definition 13.1.2. *The edge bonus is given by $LCP(G-e,S,T) - LCP(G,S,T)$, for every edge e on the lowest cost path from S to T. The path bonus is the sum of the bonuses of all edges on the path.*

Next, we define the notion of VCG overpayment. Our goal is to bound the average VCG overpayment per unit flow over all source-destination pairs. For this purpose, in the definitions of total cost and payment below, we assume a unit flow between each pair of nodes. Let V be the set of vertices in the graph G.

Definition 13.1.3. *The total cost of traffic is the sum of costs of all least cost paths:*

$$TC = \sum_{s \neq t \in V} LCP(G,S,T).$$

The total payment is the sum of all payments:

$$TP = \sum_{S \neq T \in V} \sum_{e \in LCP(G,S,T)} v(e,S,T).$$

The total bonus is the difference of total payment and total cost:

$$TB = TP - TC.$$

We may want to study several metrics—the average overpayment (and payment) per path, per unit cost, or per edge. In the case of unit costs, the last two metrics are of course the same. We define the average overpayment per unit cost, which we shall just refer to as average overpayment, to be the ratio of total bonus to total cost.

Definition 13.1.4. *The average VCG overpayment in a graph G with uniform traffic between all pairs of nodes is given by*

$$avg\ overpayment = \frac{TP - TC}{TC}.$$

The average VCG payment is the ratio of total payment to total cost, $\frac{TP}{TC} = 1 + avg\ overpayment$.

Note that there are alternative ways to measure the average overpayment. For example, we may divide for each edge the bonus over its cost, and then average over all such edges. However, with this approach it is not clear if we should first sum bonuses for a single edge which participates in several LCPs or if we should sum bonuses over each LCP, then divide by its length and average over LCPs. Each of these possibilities may give a slightly different result. We choose to work with total bonus over total cost since it is a simple and unambiguous metric to both define and measure.

Definition 13.1.5. *The average path bonus in a given graph G is the total bonus divided by the number of paths,* $\frac{TB}{\#paths}$*; similarly the average path payment is* $\frac{TP}{\#paths}$ *and the average path cost or the average distance is* $\frac{TC}{\#paths}$.

Note that the average path bonus can readily be inferred from the average overpayment TB/TC through multiplying the latter by the average distance, $TC/\#$ paths.

We need to be careful when extending the overpayment definition to a family of random graphs.

Definition 13.1.6. *The average VCG overpayment in a random graph is*

$$avg\ overpayment = \frac{\mathbf{E}[TP - TC]}{\mathbf{E}[TC]}.$$

We choose this definition instead of $\mathbf{E}[\frac{TP-TC}{TC}]$ for analytical convenience, since in the latter case the expected overpayment may be artificially blown off by a few unlikely graphs with very low total cost. In the case of Erdős-Renyi graphs with a large number of vertices n, TP and TC are well concentrated around their means so both measures give almost the same results in practice.

Figure 13.1 showed an example of a graph in which the shortest path costs 2 and the VCG payment for it is 4. In general, the VCG mechanism can pay n times higher than the cost of the shortest path. For example, consider a network with only two parallel paths, one consisting of n edges of cost 1 and the other consisting of $2n$ edges of cost 1. In this case, the first path is of lowest total cost, n; however, each edge receives a bonus of n so the extra payment to the path becomes n^2, a factor of n times larger than the true cost. Thus, the average overpayment to this single path is essentially as large as the size of the graph.

This example suggests that even short paths may receive a very high overpayment if the second best paths are much longer than the LCP. This may altogether defeat the practical use of the VCG mechanism. However, a recent calculation [39] shows that the average overpayment in the Internet graph is only 0.44 (assuming all edge costs are the same), which is incredibly low in light of the size of the Internet and the above pessimistic example. Feigenbaum *et al.* [39] explain this low overpayment by relating it to competition in the presence of network formation: as soon as a link gets a high price, a competing provider will establish another link to capture some of the revenue, thereby reducing overall payments.

13.2 VCG Overpayment in Erdős-Renyi random graphs

We begin by studying VCG overpayment in the Erdős-Renyi random graphs with n nodes and edge probability p, $G(n, p)$. Their structure and equal node degree $(n-1)p$ have been considered unrealistic in view of the recent literature on the power-law structure of small world networks, the collaboration graphs, the Internet graph, etc. [21, 120]. Nevertheless, the Erdős-Renyi graphs provide important insight for studying more general random graph models, and were used until recently to generate various network topologies [83].

In the entire section, we restrict attention to unit edge costs.

Recall that $LCP(G, u, v)$ stands for (the cost of) the lowest-cost path between u and v and whenever the graph in question is clear from the context, we shall simply write $LCP(u, v)$. Also, recall that in a random graph, the average overpayment is the ratio of expected total bonus and total cost, $\mathbf{E}[TB]/\mathbf{E}[TC]$.

Theorem 13.2.1. *For $G \in G(n,p)$ with $np = \omega(\sqrt{n \log n})$, with probability $\Omega(1 - n^{-c})$ for some constant $c > 0$, the average VCG overpayment is $\frac{p}{2-p}$.*

Proof. Our proof is based on the following two observations:

Claim 13.2.2. *For $G \in G(n,p)$ with $np = \omega(\sqrt{n \log n})$, with probability $\Omega(1 - n^{-c})$ for some constant $c > 0$, the graph G has at least two length 2 paths between every pair of vertices.*

Claim 13.2.3. *For any graph with two length-2 paths between every pair of vertices, the total bonus is m and the total cost is $n(n-1) - m$, where n and m are the number of nodes and edges in the graph.*

From here, the result follows immediately by noting that the expected number of edges in $G(n, p)$ is $n(n-1)p/2$, so the average overpayment in a random Erdős-Renyi graph is

$$\frac{\mathbf{E}[m]}{\mathbf{E}[n(n-1)-m]} = \frac{n(n-1)p/2}{n(n-1) - n(n-1)p/2} = \frac{p}{2-p}.$$

We first prove Claim 13.2.3. Assume we have a graph with two length 2 paths between every pair of vertices. Then for every pair of vertices $\{u, v\}$ there are two possibilities:

- There is an edge (u, v). Therefore $LCP(u, v) = 1$, $bonus(u, v) = 1$ since the second shortest path between u, v is of length 2, so the contribution of (u, v) to TB is 1 and to TC is 1.

- There is no edge (u, v). Therefore $LCP(u, v) = 2$, *path* $bonus(u, v) = 0$ since the second shortest path between u, v is also of length 2. So the contribution of (u, v) to TB is 0 and to TC is 2.

Therefore, if the number of edges in the graph is m, the total cost is

$$TC = 1 * m + 2 * \left(\frac{n(n-1)}{2} - m\right) = n(n-1) - m$$

and the total bonus is $TB = m$. This concludes the proof of Claim 13.2.3. Note that the average overpayment in this particular graph is given precisely by

$$\frac{TB}{TC} = \frac{m}{n(n-1)-m}.$$

We now turn to Claim 13.2.2. We will show that when $p^2 > k_c \log n/n$ for a constant $k_c = k > 0$ to be specified later, any pair of vertices in a random graph $G(n,p)$ are connected by two length 2 paths with probability $\Omega(1-n^{-c})$ for a given $c > 0$.

Consider vertices u, v. The probability that they are both connected to a given third vertex w is p^2 and so the probability that they are connected to exactly k of the remaining $n-2$ vertices is

$$\binom{n-2}{k}(p^2)^k(1-p^2)^{n-2-k}.$$

In particular, the probability that u and v are connected by at least two paths of length 2 is

$$1-(1-p^2)^{n-2}-(n-2)p^2(1-p^2)^{n-3},$$

which approaches 1 as n grows to infinity and p is constant.

Now suppose $p^2 = k\frac{\log n}{n}$. Then the probability that there are less than two length 2 paths between a given pair of nodes is

$$\begin{aligned}(1-p^2)^{n-2}+(n-2)p^2(1-p^2)^{n-3} &= \left(1-\frac{k\log n}{n}\right)^{n-2}+(n-2)\frac{k\log n}{n}\left(1-\frac{k\log n}{n}\right)^{n-3}\\ &\leq e^{k\log n}+k(\log n)e^{k\log n}\\ &= (1+k\log n)\frac{1}{n^k} \leq \frac{1}{n^{k-2}}.\end{aligned} \tag{13.2}$$

Since the probability above is a decreasing function in p, the same bound would apply for $p^2 > k\frac{\log n}{n}$. By the union bound, the probability that any of the pairs in the graph G have less than two length 2 paths connecting them is upper-bounded by

$$\binom{n}{2}\text{Prob(given pair has less than two length 2 paths)} \leq \frac{n^2}{2n^{k-2}} \leq \frac{1}{n^{k-4}}. \tag{13.3}$$

Setting $k = c+4$ for the given constant c concludes the proof of the claim and of our theorem. □

Note that a higher threshold for p would improve the probability of occurrence of two length 2 paths between every pair of vertices. On the other hand, our simulation results (see Figure 13.2) suggest that the threshold for which the formula starts to hold occurs approximately at $p = \sqrt{\frac{\log n}{n}}$.

Note that average VCG overpayment, defined as the ratio of total bonus to total cost, is also equal to the ratio of the average path bonus to average path cost (dividing the numerator and denominator by the number of vertex pairs in the graph). From Theorem 13.2.1, when $p > \sqrt{\log n/n}$,

$$\frac{TB}{TC} = \frac{\text{Avg Path Bonus}}{\text{Avg Path Cost}} = \frac{p}{2-p},$$

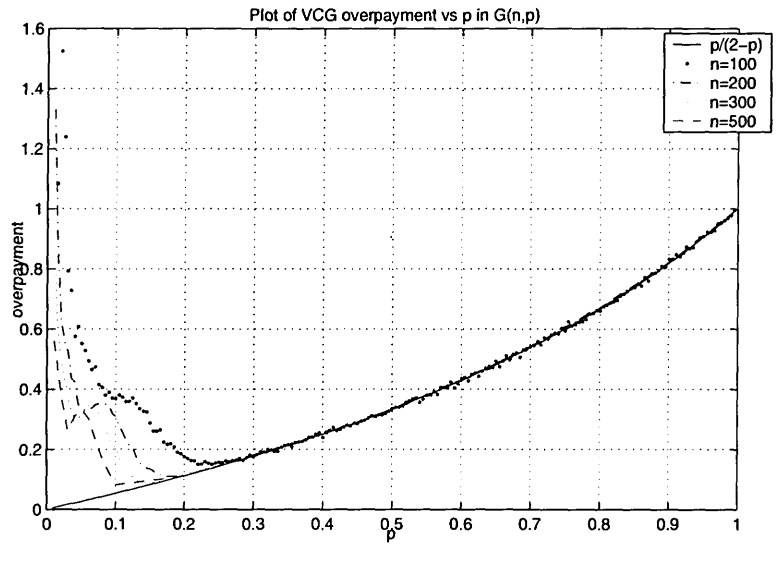


Figure 13.2. Average overpayment in $G(n, p)$.

p	Edges	TC	TC/path	TP	TP/path	TB	TB/TC
0.1	496	11011	2.22	15299	3.1	4288	0.39
0.2	1001	8962	1.81	10495	2.1	1533	0.17
0.3	1493	8407	1.70	9910	2.0	1503	0.18
0.4	2002	7898	1.60	9900	2.0	2002	0.25
0.5	2471	7429	1.50	9900	2.0	2471	0.33
0.6	3002	6898	1.39	9900	2.0	3002	0.44
0.7	3459	6441	1.30	9900	2.0	3459	0.54
0.8	3938	5962	1.20	9900	2.0	3938	0.66
0.9	4464	5436	1.10	9900	2.0	4464	0.82
1.0	4950	4950	1.00	9900	2.0	4950	1.00

Table 13.1. Average VCG overpayment for G(n,p) with 100 nodes

with high probability. When we are assured of having at least two paths of length 2 between any given pair, the average path bonus is precisely p and the average path cost is $2 - p$. This is because the average path cost is 1 with probability p (when there is an edge between the pair of nodes) and 2 with probability $1 - p$. Similarly the average path bonus is 1 with probability p and 0 with probability $1 - p$.

Consequently, the average path payment, which is the sum of the average path bonus and average path cost, is always $p + (2 - p) = 2$.

Corollary 13.2.4. *With high probability, when $p > \sqrt{\log n/n}$, the average path bonus is p and the average path payment is* 2.

Lastly, the average payment per unit cost is exactly 1 plus the average overpayment, namely $\frac{2}{2-p}$. So just like the average overpayment, average payment also rises when p does. We are now armed to explain this counterintuitive phenomenon. Whenever p is above the given threshold which guarantees the existence of paths of length 2, the bonus of each edge is upper-bounded by 1. On the other hand, in terms of overpayment, it is better to have two paths of length 2 than an edge and a second path of length 2 between a pair of vertices, since in the first case even though the lowest cost path is longer, it gets zero bonus. Similarly, the average payment per unit cost is higher because of the presence of more edges, and not because of having longer LCPs of length 2.

To appreciate these theoretical predictions, we have included Table 13.1 with simulation results for random graphs with $n = 100$ nodes. We see that initially the total payment TP and the average path payment, TP/path, fall (with the increase in competition for small values of p) and afterward they remain constant, as predicted. Also, the TC/path column, which shows the average distance, becomes equal to $2 - p$ for $p \geq 0.2$. For the same values of p, the average overpayment TB/TC, is equal to $\frac{p}{2-p}$. Note that each line in the table corresponds to a single generated random graph with 100 vertices and thus it is remarkable how well concentrated around the mean the results are. In Figure 13.2, we have plotted the average overpayment for $n = 100$, as well as for several more values of n and see that the greater the number of vertices, the greater the range of p for which the overpayment is equal to $p/(2-p)$. Also, note that $p/(2-p)$ is a lower bound on the overpayment, so together the constant upper bound from Mihail *et al.* [91], the average overpayment in $G(n, p)$ is $\Theta(1)$.

Elkind showed recently [34] that it is possible to lower the average VCG payments by deleting a subset of edges from the original network; however, it is NP-hard to determine what is the best subset of edges to delete, or even whether a given graph can benefit from edge deletion.

We note in passing that the phenomenon of reducing the VCG overpayment through deleting edges, that is, effectively through eliminating competition, is similar to Braess's Paradox for traffic flow. The latter states that sometimes closing down roads may help lower traffic delays [17].

While in general it is hard to determine the optimal set of edges to delete, in the case of $G(n, p)$ our results on overpayment and the simulation results in Fig. 13.2 suggest a simple strategy to do so.

First observe from Figure 13.2 that the lowest VCG overpayment occurs at a threshold value for p, approximately when overpayment becomes $\frac{p}{2-p}$. This threshold occurs approximately at $np = \sqrt{n \log n}$. We offer the following simple randomized algorithm. If the current average degree is below the threshold value, add edges uniformly at random until the average degree reaches the threshold and vice versa, if the current average degree is above the threshold, delete edges uniformly at random until we reduce the degree to that level.

13.3 Complete graphs with random edge costs

Once we have studied and understood well the overpayments in a random Erdős-Renyi graph with unit costs, it is natural to extend this to non-unit costs. We may ask, for example, what happens when costs are uniform on $[0, 1]$.

While we can simulate the Erdős-Renyi graph model in this case and make further observations, we shall instead look at a seemingly easier case, the complete graph with uniform costs. Cooper *et al.* [25] showed that with high probability, shortest paths in this model consist of $O((\log n)^2)$ edges and the diameter is $O(\log n/n)$.

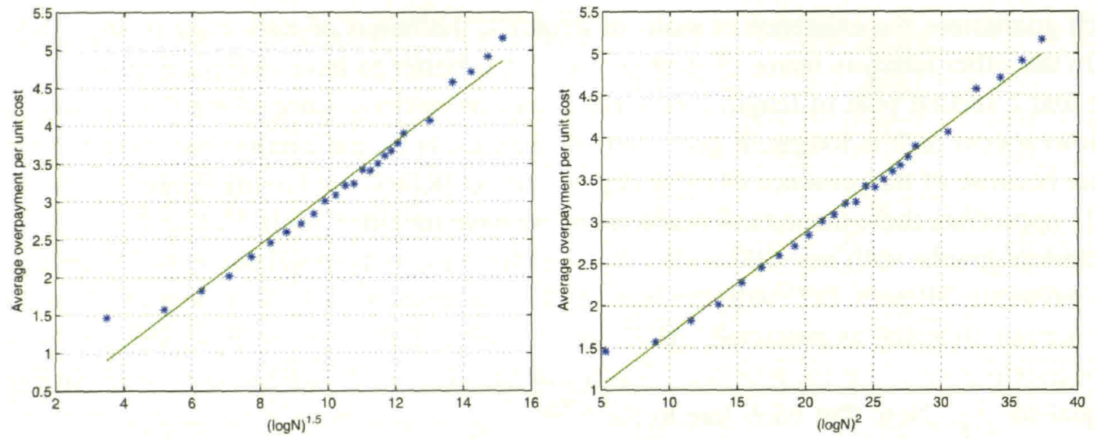


Figure 13.3. Average VCG overpayment vs $(\log N)^{\alpha}$ in complete graphs with uniformly random edge costs. Linear regression fits: (left) $-.291 + .341(\log N)^{1.5}$; (right) $-.118 + .902(\log N)^{2}$;

Note that with uniform costs it is no longer the case that payment per edge and payment per unit cost are the same.

The complete graph is in a sense perfectly competitive since it is completely connected. Thus, we may intuitively expect a low overpayment. However, just like intuition belied us before, this case turns out to be worse than even the sparse Erdős-Renyi graph with unit costs, which has overpayment of less than 1.

We simulate the overpayment in complete graphs with $n = 10$ to 450 nodes. Our results are that the average overpayment is strictly increasing with n. The complete graph with only 10 vertices has overpayment of 1.46 and the graph with 450 nodes has overpayment of 5.16, *i.e.,* over 500% of true cost. We study the rate of increase and plot the regression line fits of overpayment vs $(\log n)^{1.5}$ and $(\log n)^2$ in Figure 13.3. We also run a linear regression of the overpayment versus $(\log n)^{2.5}$ and $(\log n)^3$ and get regression coefficients $.894{+}.046(\log N)^{2.5}$ and $1.214{+}.018(\log N)^3$ respectively. It is hard to estimate the precise asymptotic behavior of the overpayment just from the generated graphs with up to 450 vertices, although, based on these regression results $\Theta((\log n)^2)$ looks like a good candidate. In any case, the overpayment seems to be lower-bounded by $\Omega((\log n)^{1.5})$

The average path cost falls a little faster than the diameter bound $O(\log n/n)$ from Cooper *et al.* [25]. For $n = 10$, the average path cost is 0.252 and it falls monotonically down to 0.015 as the number of vertices increases to $n = 450$. The average path bonus also falls monotonically from 0.329 for $n = 10$ to 0.075 for $n = 450$, although it falls at a slower rate than the average path cost.

13.4 Power-law graphs

Perhaps the most common real world networks are scale-free, that is their degree sequence follows a power-law distribution: $\mathbf{Pr}(\text{degree} = d) \propto 1/d^{\beta}$, where typically $2 < \beta < 3$. The Internet graphs have β between 2.1 and 2.45 [120], while the collaboration graph of *Mathematical Reviews* has $\beta = 2.97$ [21].

Chung and Lu have pioneered the study of random graphs with given expected degrees and

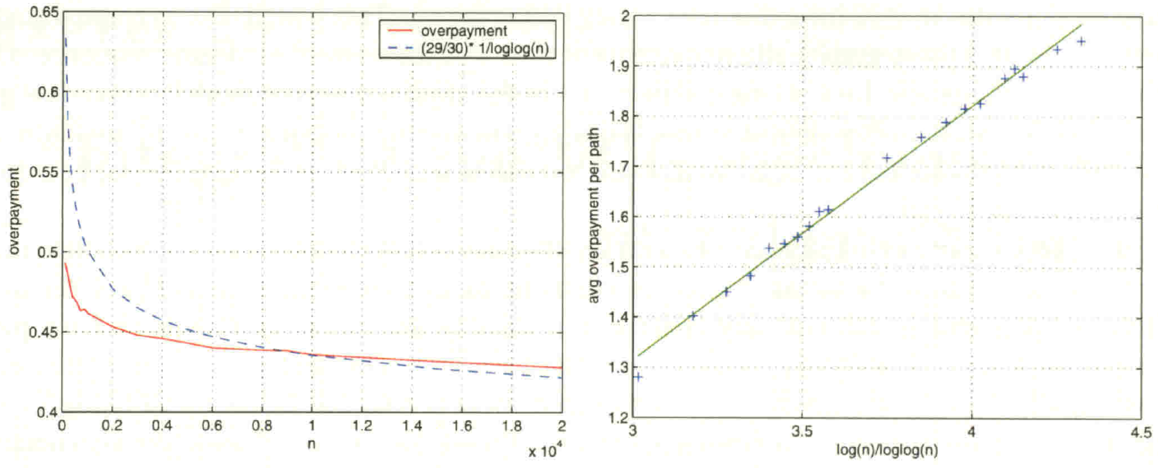


Figure 13.4. Power-law graphs with unit costs. Left: Average VCG overpayment, lower-bounded by $\Theta(\log\log n)$. Right: Average path bonus seems to grow more slowly than $\Theta(\log n/\log\log n)$.

have shown that the average distance of a power law random graph with exponent $2 < \beta < 3$ is $O(\log\log n)$. The average distance changes to $\Theta(\log n/\log\log n)$ when $\beta = 3$ and to $O(\log n)$ when $\beta > 3$ [21]. Mihail, Papadimitriou and Saberi concluded that, as an immediate corollary of Chung and Lu's result, the average VCG overpayment is bounded by $O(\log\log n)$ when $\beta < 3$ [91]. They also state the conjecture that the overpayment in this case is constant.

The other popular model for generating scale-free graphs is the preferential attachment model, which generates graphs by connecting each new vertex to the already existing vertices with probabilities proportional to their current degrees. Bollobas and Riordan [13] offer a precise definition of the model, which was first suggested by Barabasi [6].

The standard preferential attachment model has a parameter M, which stands for the number of vertices that each newly arriving vertex attaches to. This parameter determines the average degree. Since each new vertex adds M new edges, the total number of edges in the graph is approximately Mn, where n is the number of nodes. If the average degree is d, the total number of edges is $nd/2$ since each edge is counted twice. Thus, $d \approx 2M$.

The average degree in the Internet AS-level graph (essentially, the graph of Internet Service Providers) from September 2003 is 4.12, its maximum degree is 2415 and its power-law exponent is about 2.17 [116]. Based on the average degree, it is most appropriate to simulate the graph with the preferential attachment model with $M = 2$. Note, however, that it is impossible to match all parameters such as average degree, power-law exponent of the degree sequence, maximum degree, etc. Bollobas *et al.* have shown that, regardless of M, the preferential attachment model generates a degree sequence with a power-law distribution with exponent 3 [14]. Despite these differences from the real-world network properties, we base our simulations on the preferential attachment model with $M = 2$ since it is easy to generate graphs and also to obtain their biconnected components simply by pruning the vertices of degree 0 and 1. In our results, we state the original number of nodes n that we simulate; however, the overpayment and other metrics are based on the biconnected component, which contains about 80% of the original vertices.

Bollobas and Riordan [13] proved that power-law graphs generated by the preferential attachment model with $M \geq 2$ have diameter $O(\log n / \log\log n)$. This bounds the average distance as well. Recall that these graphs all have exponent $\beta = 3$ of the power-law degree sequence. On the other hand, Chung and Lu [21] have shown that in the expected degree model, power-law graphs with $\beta = 3$ have diameter almost surely $\Theta(\log n)$ and average distance $\Theta(\log n / \log\log n)$. Our simulation results for the average distance are consistent with both models, as they appear to grow as $\Theta(\log n / \log\log n)$.

It is remarkable that even though the average distance is higher in this case ($\beta = 3$) than the case $2 < \beta < 3$, in which the distance is $O(\log\log n)$ in the expected degree model [21], the average VCG overpayment is still not only constant but steadily declining. In Figure 13.4 we plot the overpayment, as well as $\frac{29}{30} * 1/\log\log n$ for a comparison; we see that $\log\log n$ seems to decrease more rapidly than the overpayment, so the overpayment is bounded below by $\Omega(1/\log\log n)$. In addition, the overpayment is bounded above by 0.5 for all n in our simulation. We also obtain that the average path bonus is increasing with n. For example, when $n = 100$, the average path cost is 2.59 and the path bonus is 1.28. They increase to 4.56 and 1.95 respectively when $n = 20,000$. Naturally, the average distance increases with n and, as the paths get longer, they receive higher payments and, as it turns out, higher bonuses.

Recall that the average overpayment and the average path bonus are related as

$$\frac{TB}{TC} = \frac{\text{avg path bonus}}{\text{avg path cost}},$$

that is, average overpayment is equal to the average path bonus divided by the average distance. Since the average distance is $O(\log n / \log\log n)$ and the average overpayment TB/TC is $O(1)$, then the average path bonus is also $O(\log n / \log\log n)$. This is just an upper bound, and is consistent with the plot on the right in Figure 13.4. If the average distance is exactly $\Theta(\log n / \log\log n)$, then by our estimates above the average overpayment is asymptotically lower bounded by $1/\log\log n$ and so the average path bonus is $\Omega(\log n / (\log\log n)^2)$.

Lastly, we consider power-law graphs with Uniform Random Costs. Recall that in complete graphs, the overpayment is exactly 1 with unit costs and exceeds $\Theta((\log n)^{1.5})$ with uniformly random costs, thus the cost distribution alone can exacerbate payments significantly.

Somewhat surprisingly, we obtain that the average overpayment in our simulated power-law graphs with uniformly random costs remains 0.85 ± 0.01 as n increases from 100 to $4,000$ so it seems to be constant for any n.

These results suggest that the average VCG overpayment is bounded by a constant in power-law graphs, regardless of the value of β, and regardless even of whether edge costs are unit or uniformly random. It is an intriguing open question to prove this theoretically, for either the preferential attachment or expected degree model, under general cost distributions.

In conclusion to this chapter, our results suggest that for common real-world networks the VCG payments will not be too much higher than the true costs. Hence VCG-related mechanisms should remain a viable option for integrating prices and incentives into the Internet protocols and in other real-world networks.

CHAPTER 14

Integrating the VCG mechanism with inter-domain routing

In the previous two chapters, we assumed that each edge in the network is owned by an independent selfish agent with a fixed per packet cost. This invites two questions: 1) how can the mechanisms be incorporated into the current routing protocols? 2) how can we extend this to a more realistic model where an agent's utility is affected by more than one edge?

Our first-price mechanisms from Chapter 12 lead to lower prices but more complex strategic behavior: in some cases equilibria did not even exist, and in cases in which they did, it remained open how an equilibrium can be implemented or reached. Consequently, this may translate to an unstable routing protocol and one needs to investigate equilibrium implementation further before proposing changes to the protocols. On the other hand, the VCG allocation, payments and optimal strategic behavior are well understood and incorporating them into the Border Gateway Protocol (BGP)—the protocol for communication between autonomous systems, has been addressed in the simple model above. In particular, it was noticed [39] that social welfare can be optimized in routing, if one assumes that each Autonomous System (AS)[1] has a fixed per packet cost, via the VCG mechanism with payments; and in fact, that this can be achieved in a way that is very "BGP-friendly," *i.e.*, can be implemented by minimal disruption of BGP's ordinary operation.

In a subsequent paper [41], the problem of more realistic BGP routing was addressed in the same spirit. Each AS was assumed to have a utility for each path to the destination (assumed in this literature to be a fixed node 0), and the goal was to maximize total utility. It was shown that the problem is too hard to solve in general, even with no consideration to incentive compatibility, while a special case, in which the utility of a path only depends on the next hop, is easy to solve in an incentive-compatible way but hard to implement on BGP. To show this latter negative result, Feigenbaum *et al.* [41] formalize what it means for an algorithm to be "BGP-friendly": roughly speaking, a local distributed algorithm with quick convergence, small storage needs, and no rippling updates in case of small parameter changes. All said, the message of Feigenbaum, Shenker and Sami [41] was that, if one models BGP routing a little more realistically, incentive compatibility becomes problematic. This negative message was ameliorated in Feigenbaum *et al.* [40] where it was pointed out that, if one further restricts the special case of next-hop utilities so that paths are required to be of a particular kind mandated by the kinds of inter-AS agreements seen in practice,

[1]"Within the Internet, an autonomous system (AS) is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators that presents a common, clearly defined routing policy to the Internet (cf. RFC 1930, Section 3)." [129]

called *valley-free* here, BGP-friendly incentive compatibility is restored.

To answer both questions 1) and 2) above, in this chapter we present an elementary model of BGP routing. The key feature of our model is that *path preferences are based exclusively on per packet costs and per packet agreed-upon compensation* between *adjacent* nodes. In other words, we look into the utilities of each path to each AS, taken as raw data in previous literature, and postulate that they are linear functions of the traffic, depending on two factors: objective per packet costs to each AS for each incoming or outgoing link, and agreed per packet payment, positive or negative, to the AS for this link and direction. As a result, social welfare optimization becomes a min-cost flow problem, and incentive-compatibility can always be achieved (via VCG) in polynomial time. If there are no capacity constraints, we show (Theorem 14.4.1) that the resulting algorithm is BGP-friendly, essentially because the BGP-friendly version of the Bellman-Ford algorithm in [39] can be extended to cover this case. When capacities are present, the algorithm becomes a more generic min-cost flow computation (Theorem 14.4.2), and, as we show by a counterexample, does not adhere to the criteria of BGP-friendliness (it may not converge fast enough), even though it is still a local, distributed algorithm with modest memory requirements and no need for global data. If, on top of this, we also require that the paths be of the "valley-free" kind suggested by the kinds of agreements between ASes one sees in practice (that is, the kind of restriction which led to tractability in [40]), the resulting algorithm solves a rather generic linear program (Theorem 14.4.3), and so local, distributed computation appears to be impossible.

14.1 Basic model

We model interdomain routing as a symmetric directed network with node set $V = \{0, 1, ..., n\}$ and edges E, where node 0 is assumed to be a fixed *sink* (the destination of all packages, assumed unique as is common in this literature). We postulate that the network is symmetric in that if $(i, j) \in E$ then also $(j, i) \in E$. There are no self-loops. Each node i has a *demand* of k_i packets it wants to send to the sink. In addition, each node i has a *per packet value* $v_{i,e}$ (sometimes also denoted as $v_i(e)$) for each of its incident edges e, and a value π_i for each of its packets that gets delivered. The *cost* of an edge $e = (i, j) \in E$ is the negative of the sum of values of i and j for it, $p_e = -(v_{i,e} + v_{j,e})$.

We denote by θ_i the *type* of node i, that is the collection of values for its incident edges and its value per packet delivery. Denote by θ the vector of all node types and by θ_{-i} the vector of all node types except that of node i.

If F is an integer-valued flow through this network, with sink 0 and sources at all other nodes with the given demands, then the welfare of each node $i \neq 0$ from this flow is

$$w_i(F, \theta_i) = \sum_{e:\ i \in e} v_i(e) F(e) + \pi_i F_i,$$

where by $F_i = \sum_j F(i, j) - \sum_j F(j, i)$ we denote the flow out of i, assumed to be at most k_i. The total welfare of all nodes under flow F is

$$W(F) = \sum_{i \in V \setminus \{0\}} \pi_i F_i - \sum_{e \in E} p_e F(e).$$

Let $F^*(\theta)$ be the welfare-maximizing flow for types θ, and let $F^*_{-i}(\theta^{-i})$ is the welfare of the optimum flow when node i is deleted from the network. We assume initially that all capacities are infinite, which implies that the optimum flow is the union of n or fewer flow-weighted source-to-sink shortest paths; this assumption is reconsidered in Section 14.2.

14.1.1 VCG mechanism

Notice that in order to compute the optimum flow we need to know the types of all players; the difficulty is, of course, that the type of player $i > 0$ is known only to player i, who is not inclined to publicize it in the absence of appropriate incentives. The *VCG mechanism* for this problem incentivizes the players to reveal their true types, and thus participate in a socially optimum flow, by making payments to them. Consider, in particular, the following transfers for each node (negative for payments made by the node and positive for payments received by the node).

$$\begin{aligned} t_i(\theta) &= \Big[\sum_{j\neq i} w_j(F^*(\theta),\theta_j)\Big] - \Big[\sum_{j\neq i} w_j(F^*_{-i}(\theta_{-i}),\theta_j)\Big] \\ &= \Big[\sum_{\substack{j\neq i\\ \pi_j\geq P_j}} k_j\cdot(\pi_j - P_j^{-i})\Big] - \Big[\sum_{\substack{j\neq i\\ \pi_j\geq P_{j,-i}}} k_j\cdot(\pi_j - P_{j,-i})\Big], \end{aligned} \tag{14.1}$$

where P_j is the cost of the cheapest path from j to 0; $P_{j,-i}$ is the cost of the cheapest path from j to 0 which does not go through node i. P_j^{-i} is the cost of the cheapest path $path_j$ from j to 0 without taking costs potentially incurred by i into account: if $i \notin path_j$, $P_j^{-i} = P_j$, otherwise $P_j^{-i} = P_j + (v_{i,e_1} + v_{i,e_2})$ where $e_1, e_2 \in path_j$ denote the edges incident to i. Note that nodes send their own packets only if they have a nonnegative welfare for doing so, namely if the per packet delivery value π_j is at least as big as the path cost P_j to the destination. Thus, they send either zero or all k_j of their packets.

The proof that these transfers lead to truthful reporting is analogous to the corresponding proof about the Groves mechanism in Mas-Collel *et al.* [87] specialized to the current situation. We include it here for completeness:

Theorem 14.1.1. *When the nodes are selfish agents and have values for adjacent edges and for the delivery of their own packets, there is a strategyproof pricing mechanism under which the lowest-cost paths are chosen, and the payments are of the form given in Equation (14.1).*

Proof. Suppose truth is not a dominant strategy for some node i, *i.e.*, the node gets higher utility by reporting a collection of values $\hat{\theta}_i$ different from his true values θ_i when the other nodes report θ_{-i}. The utility of the node is its welfare plus the payment to the node by the mechanism:

$$w_i(F^*(\hat{\theta}_i,\theta_{-i}),\theta_i) + t_i(\hat{\theta}_i,\theta_{-i}) > w_i(F^*(\theta),\theta_i) + t_i(\theta_i,\theta_{-i}).$$

Substituting the form of the transfer on both sides and canceling identical terms, we get

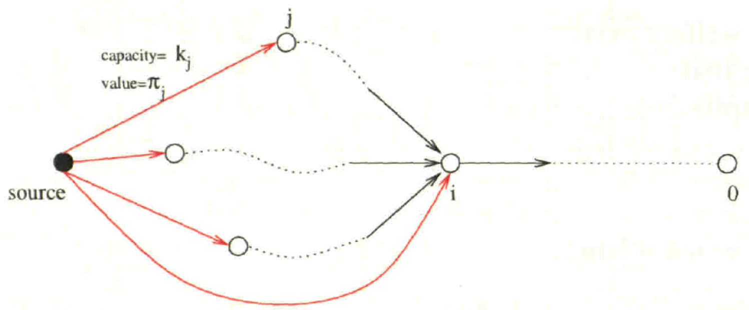


Figure 14.1. In a model with capacities, we add a supersource and edges from it to each node j, of capacity k_j and cost $-\pi_j$ (equivalently, value π_j). The socially optimal solution corresponds to the min-cost flow on the induced graph.

$$
\begin{aligned}
& w_i(F^*(\hat{\theta}_i, \theta_{-i}), \theta_i) + \Big[\sum_{j \neq i} w_j(F^*(\hat{\theta}_i, \theta_{-i}), \theta_j)\Big] > w_i(F^*(\theta), \theta_i) + \Big[\sum_{j \neq i} w_j(F^*(\theta), \theta_j)\Big] \\
\Leftrightarrow \quad & W(F^*(\hat{\theta}_i, \theta_{-i}), \theta) > W(F^*(\theta), \theta).
\end{aligned}
$$

The last inequality contradicts the fact that $F^*(\theta)$ is the welfare maximizing choice of paths (*i.e.,* the least cost paths) for node types θ. $\square$

14.2 Model with capacities

In this subsection we consider the same basic model, with the addition that each edge e has a corresponding capacity c_e. We would like to find a min-cost (multicommodity) flow from all nodes to the sink 0, satisfying the demands of the nodes. We can transform the problem to an equivalent one by adding a new node—a supersource, which is connected to each node j via an edge of cost $-\pi_j$ and capacity equal to the demand k_j at node j (see Figure 14.1).

Denoe the resulting min-cost flow with known types θ by $F^*(\theta)$, and the min-cost flow in the graph with node i removed by $F^*_{-i}(\theta_{-i})$. We can now get a VCG mechanism similar to the one in the basic model. As before, the total welfare is

$$
W(F^*(\theta), \theta) = \sum_i w_i(F^*(\theta), \theta_i)
$$

where $w_i(F^*(\theta), \theta_i)$ is the welfare of the flow from i (more precisely, from the supersource through i) to 0. Similarly, the VCG mechanism is specified by the transfers

$$
t_i(\theta) \quad = \quad \Big[\sum_{j \neq i} w_j(F^*(\theta), \theta_j)\Big] - \Big[\sum_{j \neq i} w_j(F^*_{-i}(\theta_{-i}), \theta_j)\Big]
$$

and a proof of correctness (truthfulness) of the mechanism follows as before.

14.3 Model with economic relationships

The economic relationships between individual ASes in the Internet severely influence the paths which can be taken in the BGP graph. So far, we assumed that all paths which are present in the underlying undirected graph (there is an edge between two ASes, if they are connected by a physical link) are valid. In reality, this is not the case. Routing policies which are based on the economic relationships between ASes forbid many paths which theoretically exist. Inferring these economic relationships and investigating the resulting consequences for the connectivity of the BGP graph have attracted a large amount of scientific interest recently, see, e.g., [1, 7, 31, 35, 48, 55, 122].

Below we shall give a detailed description of the valley-free path model which classifies the prohibited paths in the BGP graph.

The Valley-Free Path Model. In this model there are basically three different types of relationships a pair of connected ASes can be in: either *customer-provider*, in which the customer pays the provider to obtain access to the Internet, or *peer-peer*, in which both peers agree on mutually routing traffic of their customers for each other free of charge, or *sibling*, in which both siblings agree on mutually routing any traffic for each other free of charge. Note that an individual AS may take several roles—as customer, provider, sibling, or peer—simultaneously; for instance, it can be a customer of one AS and at the same time a provider for another AS.

In the following, for ease of exposition we shall focus on customer-provider relationships only. The other types of relationships (peer-peer, sibling) can be incorporated easily, as we shall note in Section 14.4.3.

We call a directed graph $G = (V, E)$ a *ToR graph*, if it contains no self loops and the edge directions describe the economic relationships. If the AS i is a customer of a provider AS j, we direct the edge between i and j towards j. This follows the terminology of [122].

In practice routing is done in the following way. If AS j is a provider of i (*i.e.*, $(i, j) \in E$), it announces all its routes to i, but AS i on the other hand, only announces its own routes and the routes of its customers to j. The idea behind this is that i pays j for the connection and thus is not inclined to take over "work" for j. This would happen if i also announced the routes it has from other providers. Then it would potentially have to donate bandwidth to packets that arrive from the provider j, only to proceed to another provider. This clearly is of no benefit to i itself or to any of its customers. From its point of view, j should find a different route for the corresponding packets.

This leads to the model proposed in [122] that a path is *valid* if and only if it consists of a sequence of customer-provider edges ($\bullet\!\!\longrightarrow\!\!\bullet$) followed by a sequence of provider-customer edges ($\bullet\!\!\longleftarrow\!\!\bullet$). The first part, containing only customer-provider edges, is called the *forward part* of the path. The last part, containing only provider-customer edges, is called the *backward part* of the path. It is easy to see that the following is an equivalent definition of the validity of a path:

> A path with edges $\{e_1, e_2, \cdots, e_r\}$ in the ToR graph G is a *valid path* in G, if and only if there is no inner node in the path for which its incident edges e_{i-1} and e_i are both outgoing from the node.

If such an inner node—one which does have this property—exists, it is called a *valley*. The intuition behind this name is that outgoing edges point "upwards", out of the valley. In the literature the situation that a path contains a valley is also called an *anomaly*. A flow which only uses valley-free paths we call a *valid* or *valley-free* flow.

The VCG mechanism. The transfers can be specified as in Section 14.2 for the model with capacities. The only difference is that all flows (*i.e.*, F^* and F^*_{-i} for all i) must be valley-free (and may be fractional).

14.4 Distributed computation of VCG payments

It is of great interest to determine to what extent the payments $t_i(\theta)$ can be computed not only efficiently, but in a distributed manner which is "BGP-friendly," that is, compatible with current usage of the BGP protocol. In [41] this concept of "BGP-friendliness" was formalized as three requirements:

1. The algorithm should converge in a number of rounds that is proportional to the diameter of the graph, and not its size.

2. Only local data should be needed.

3. No rippling updates should be needed as data changes.

Here we relax requirement (1) to a number of rounds that is proportional to the diameter times R, where R is the ratio between the largest and smallest edge cost. This is necessary (also in [41], where the stricter version is used by oversight) because the computed shortest path, whose length is the upper bound on convergence time, may be longer than the diameter. We do not bother to formalize here the second and third requirement (the reader is referred to [41]) because our algorithms either trivially satisfy any conceivable version, or fail to satisfy (1). As it turns out, this important aspect of BGP-friendliness sets the basic model apart from the model with capacities. In both cases the implementation is quite simple and makes only modest use of local resources. But only in the former case the strict conditions on the convergence time are fulfilled.

14.4.1 Basic Model

For the basic model, it is easy to adapt the approach presented by Feigenbaum *et al.* [39]. BGP is a *path-vector* protocol which computes the lowest-cost paths (LCPs) in a sequence of stages. In a stage each node in the network sends all the LCPs it knows of to its neighbors. It also receives LCPs from its neighbors. If these contain shorter paths than the ones it has currently stored, it updates the list of its own LCPs. This basically corresponds to a distributed computation of all shortest paths via the Bellman-Ford algorithm. The computation terminates after d stages and involves $O(nd)$ communication on any edge, where d denotes the maximum number of edges on an LCP.

Let $\text{diam}'(G)$ denote the maximum diameter of $G \setminus \{i\}$ over all nodes i for which $G \setminus \{i\}$ is still connected.

Theorem 14.4.1. *In the basic per packet utility model without capacity constraints (described in Section 14.1), the Vickrey-Clarke-Groves allocation and payments can be computed in a distributed, BGP-friendly manner. The computation converges in $O(diam'(G) \cdot R)$ rounds of communication, where R is the ratio between the largest and smallest edge cost.*

Proof. Our proof follows analogously to that of Theorem 2 in Feigenbaum *et al.* [39]. The authors there propose an extension of the path-vector protocol which computes not only the lowest cost paths, but at the same time the lowest cost paths which do not traverse a given node i. These two quantities are then used to compute the payments for node i. The computation of paths avoiding node i increases the number of stages and communication needed to d' and $O(nd')$ respectively. Here d' denotes the maximum number of edges on an LCP avoiding node i, over all nodes i for which $G \setminus \{i\}$ is still connected. Feigenbaum *et al.* argue that this is still an acceptable convergence time.

The only difference of the approach in [39] to ours is that the per packet values in our model are given individually for each edge and node, *i.e.*, as $v_{i,e}$, and not only as one total value per node.[2] Hence, it is easy to adapt their method to compute the values P_j and $P_{j,-i}$, for $j, i \in \{1, \dots, n\}$, which is all we need to compute the VCG payments

$$t_i(\theta) = \Big[\sum_{\substack{j \neq i \\ \pi_j \geq P_j}} k_j \cdot (\pi_j - P_j^{-i}) \Big] - \Big[\sum_{\substack{j \neq i \\ \pi_j \geq P_{j,-i}}} k_j \cdot (\pi_j - P_{j,-i}) \Big].$$

Note that the partial path cost P_j^{-i} can be easily derived from the cost P_j of the cheapest path from j to the sink.

Since $d' \leq \text{diam}'(G) \cdot R$, we get the desired running time. □

14.4.2 Model with capacities

Instead of lowest cost paths with and without node i, we now need to know a min-cost flow $F(\theta)$ and a min-cost flow $F_{-i}(\theta)$ avoiding node i for each of the payments $t_i(\theta)$, $i \in \{1, \dots, n\}$. In the following, we shall explain how to compute $F(\theta)$ in a distributed fashion. The flow $F_{-i}(\theta)$ can be computed correspondingly by blocking node i. Therefore, altogether $(n + 1)$ flow computations are performed, one for $F(\theta)$ and n for the $F_{-i}(\theta)$, $i \in V \setminus \{0\}$.

We assume the sink 0 controls all the computations: it chooses which node is blocked (in the $F_{-i}(\theta)$ case), it selects paths to send flow along together with the corresponding amounts, and it recognizes when a min-cost flow computation is finished. These all are computationally simple tasks. The only intensive computations needed will be those to obtain the shortest paths with respect to certain costs and where certain edges may be blocked. These will be done in a distributed manner applying the standard distributed Bellman-Ford algorithm, which is used by BGP as mentioned above.

Distributed Computation of $F(\theta)$. We start with a description of a simple Ford-Fulkerson approach [43] of computing a min-cost flow from the supersource to the sink via augmenting shortest paths. Then we explain how to modify it to use the Edmonds-Karp scaling technique [33].

[2]This allows for more fine-granular and thus more realistic modeling.

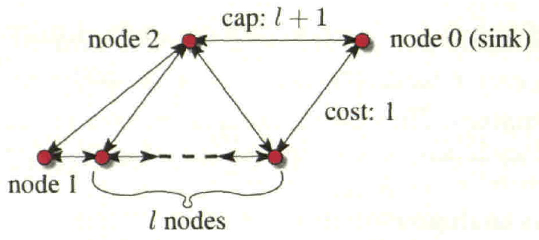


Figure 14.2. All edges have capacity 1, except the top edge with capacity $l+1$. The edge costs are all 0, except the rightmost edge with cost 1. All nodes have a demand of 1 to be sent to node 0.

A virtual residual graph is laid over the given network. The residual edge capacities and costs are derived from the original graph. The residual capacities depend on the flow present on the corresponding residual edges and thus may change during the computation of the flow. Each node keeps track of flow values on residual edges that are incident to it.

Consider an original pair of directed edges (i,j) and (j,i) with costs $p_{(i,j)}$ and $p_{(j,i)}$. We assume the costs to be greater than or equal to 0. Let $f_{(i,j)}$ and $f_{(j,i)}$ denote the flow amounts on these edges, only one of which may be greater than 0. Otherwise, a circular flow of $\min(f_{(i,j)}, f_{(j,i)})$ is subtracted from both without increasing the costs. The residual capacities are set to $c'_{(i,j)} = c_{(i,j)} - f_{(i,j)}$ and $c'_{(j,i)} = c_{(j,i)} - f_{(j,i)}$. Additionally, we add the virtual edges $(i,\bar{j})$ and $(\bar{j},i)$ with capacities $c_{(i,\bar{j})} = f_{(j,i)}$ and $c_{(\bar{j},i)} = f_{(i,j)}$ and costs $p_{(i,\bar{j})} = -p_{(i,j)}$ and $p_{(\bar{j},i)} = -p_{(j,i)}$. Flow sent onto these edges is subtracted from the corresponding flow on the edge in the opposite direction. Finally, for each $i \in V \setminus \{0\}$ a virtual edge is added from the supersource to node i with cost $-\pi_i$.

The algorithm now proceeds as follows (steps 2-4 comprise a phase):

1. For each node $i \in V$, initialize the flow values $f_{(i,j)} = f_{(j,i)} = 0$ of all incident residual edges. Update the local capacities as described above.

2. Compute the shortest paths in the current residual graph only considering edges with capacities greater than 0. Do this with the distributed Bellman-Ford algorithm, adapting the BGP implementation. Modify the algorithm to also forward the bottleneck capacity of each path.

3. The sink checks the min-cost path to the supersource. If the cost is 0, we are done. Otherwise, send a flow corresponding to the bottleneck capacity along the path. This is done by sending a message along the path, which notifies the contained nodes to update their local flow values (and thus capacities).

4. Continue at step 2 with the updated residual graph.

Theorem 14.4.2. *In the per packet utility model with capacity constraints (described in Section 14.2) the Vickrey-Clarke-Groves allocation and payments can be computed in a distributed manner. The computation converges in $O(n^3 \cdot \log C)$ rounds of communication, where $C = \max\{c_e | e \in E\}$ is the maximum edge capacity.*

Proof. Each phase consists of a (re)computation of the shortest paths in Step 2. Note that in the capacitated case the number of rounds of communication for a shortest paths computation is not bounded by d or d' anymore. It may actually take up to n rounds of communication, as the example in the paragraph below shows.

The algorithm finishes in $O(|E| \cdot C)$ phases. This can be improved to $O(|E| \cdot \log C)$ by applying the well-known scaling technique. To this end, a variable Δ is introduced and initialized to $2^{\lceil \log C \rceil - 1}$ in step 1. In step 2, only edges with capacity Δ are considered. In step 3, Δ is updated to $\Delta/2$, if no more negative cost paths are found (unless $\Delta = 1$: then we are done). The updated Δ is broadcast to all nodes.

As mentioned, with $(n+1)$ flow computations we can calculate all node payments $t_i(\theta)$, which yields the desired number of rounds $O(n^3 \cdot \log C)$. $\square$

Shortest Paths Computation. With capacities, the number of rounds of communication to compute the shortest paths cannot be bounded by d (or d') anymore. Figure 14.2 shows a counterexample where the shortest path in the residual graph has length $n-2$, whereas the number of hops in the corresponding LCP in the original graph is 2. Assume that all nodes have already (virtually) sent their flow through the residual graph except node 1, which is selected last. Since the nodes are indistinguishable, we may assume this. The only path remaining in the residual graph is the one at the bottom of length $n-2$, since the capacities of all other edges (expect $(1,2)$) are fully saturated by flow sent to the sink via node 2. This compares to the LCP with only two edges from node 1 over node 2 directly to node 0.

14.4.3 Model with economic relationships

In the following, we shall explain the two-layer graph, a helpful notion which was originally suggested in [35]. With the help of the two-layer graph it will be easy to see that one can compute min-cost valley-free flows as needed in our model with capacities introduced in Section 14.2.

The Two-Layer Model. From a ToR graph $G = (V, E)$ with source s and sink $t \in V$ we construct a directed *two-layer graph* H, in the following way (see Figure 14.3 for an example): Two copies of the graph G are made, called the *lower* and the *upper layer.* In the upper layer all edge-directions are reversed. In addition, every node i in the lower layer is connected with an edge to the corresponding copy of i, denoted i', in the upper layer. The edge is directed from i to i'. Finally, we obtain the two-layer graph H by merging the two s-nodes (of lower and upper layer) and also the two t-nodes, and by removing the incoming edges of s and the outgoing edges of t.

A valid path $path_G = \{i_1 \cdots i_r\}$ in G with $i_1 = s$ and $i_r = t$ is equivalent to a directed path in H in the following way: The forward part of $path_G$, that is the part containing all edges $(i_q, i_{q+1}) \in path_G$, is routed in the lower layer. Then, there is a possible switch to the upper layer with a (i, i')-type edge (there can be at most one such switch for each path). The backward part of $path_G$ is routed in the upper layer. In other words, for each original edge $(i_{q+1}, i_q) \in path_G$, the corresponding edge (i'_q, i'_{q+1}) of the upper layer is traversed. If there is only a forward (respectively backward) part of $path_G$, then the corresponding path in H is only in the lower (respectively upper) layer.

This definition of the two-layer graph can easily be extended to the case of multiple sources. Note that a peer-peer relationship between two nodes $i, j \in V$ can be incorporated by adding the edges (i, j') and (j, i') from the lower to the upper layer (reflecting that at most one peer-peer edge

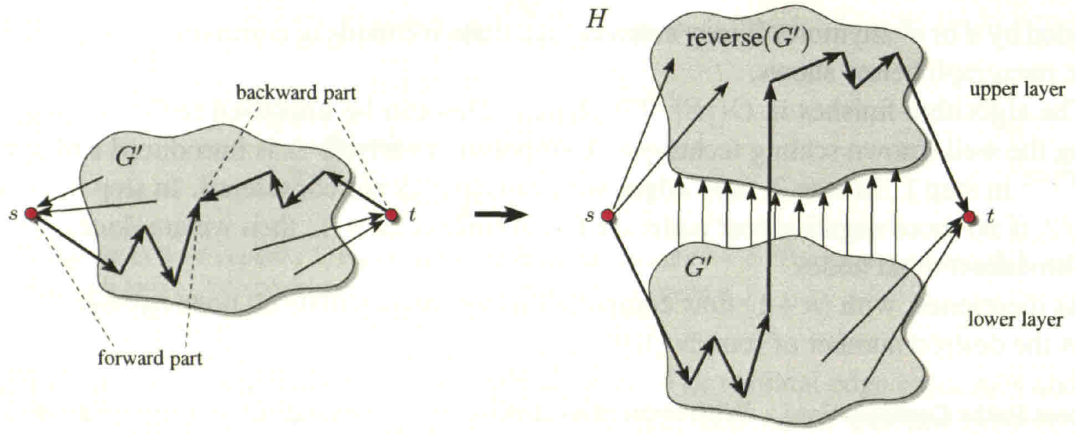


Figure 14.3. A path in the ToR graph G and the corresponding path in the two-layer graph H. (G' is G, excluding s and t.)

is allowed between the forward and the backward part of a path). Similarly, a sibling relationship between two nodes $i, j \in V$ can be incorporated by adding the symmetric edges (i, j), (j, i), (i', j'), and (j', i') in both layers (reflecting that sibling edges are allowed at arbitrary points in a path).

Min-Cost Valley-Free Flows. By simply computing a min-cost flow in the two-layer graph, it is easy to derive a valley-free flow, which will have at most the cost of an optimum min-cost valley-free flow. The edge capacities may be violated by at most a factor of two though, since each edge may be used twice: once in the upper and once in the lower layer. Note that such a min-cost flow could be computed in a distributed fashion by slightly modifying the approach described in Section 14.4.2. Unfortunately, this approximate solution cannot be used to compute the VCG allocation and payments, because the truthfulness of the mechanism holds only for the exact form of payments from the optimal solution. Thus, we need to compute the optimal solution.

Theorem 14.4.3. *In the per-packet utility model with capacity constraints and economic relationships (described in Section 14.3), the Vickrey-Clarke-Groves allocation and payments can be computed in polynomial time with a linear programming-based approach.*

Proof. For a given graph, consider its corresponding two-layer graph described above. Then the exact allocation and payments in the original graph can be computed with the help of a standard linear programming flow formulation on the two-layer graph, with added constraints to bound the joint flow on the upper and lower layer edges. In particular, for each edge $(i, j) \in E$ in the original ToR graph, we add a joint capacity constraint for (i, j) and (j', i') in the two-layer graph. □

Note that the existence of an exact algorithm based on augmenting paths seems unlikely. Usually, for integral capacities such algorithms aim at computing an optimal integral solution, *i.e.,* for unit capacities a solution would consist of edge-disjoint paths. However, computing the maximum number of disjoint valley-free paths between two nodes s, t is inapproximable within a factor of $(2 - \epsilon)$, unless P = NP [35].

14.5 Concluding remarks and open problems

Despite the fact that incentive compatibility for BGP routing had been known to be problematic in general, as well as for several apparently realistic special cases, we have identified one important special case of practical importance, namely the one in which path utilities depend on local per packet costs, as well as delivery values. In this case, incentive compatibility is achievable through payments which can be computed efficiently and in a BGP-compatible way. Adding capacities and the "valley-free" constraint for paths makes incentives harder to compute in a BGP-compatible way, but still tractable.

Regarding the latter point, in this work we have simply pointed out that the algorithms we devised for VCG incentive computation are not implementable in a BGP-compatible way. It would be interesting to actually prove that this is inherent to the problem, that is, to prove a lower bound on the convergence time of any algorithm solving the min-cost flow problem and its valley-free constrained case.

Our model for path utilities is suggestive of a more general project for understanding BGP routing: We postulate that each directed edge in and out of every node has a value for this node, depending on the cost to this node, as well as agreed upon payments to or from its neighbors, for each packet sent or received along this edge. Suppose that the graph, the demand and the per packet cost and delivery values of each node are given. These induce a game in which the strategies are payment agreements between neighbors, and the utility to each node is the one obtained by our model of BGP min-cost routing. This game is a very realistic network creation game, with special emphasis on BGP routing. The quality of equilibria compared to the social optimum (*i.e.*, the price of anarchy and its variants) for this game would be a most interesting research direction. The social optimum is, of course, the min-cost flow with only costs and delivery values taken into account. Further, such a model would allow one to study how inter-AS agreements can depend on the underlying fundamentals of each AS such as costs, delivery value, demand, and position in the network.

Bibliographic notes Chapter 12 is joint work with Nicole Immorlica, David Karger, Rahul Sami [69]; Chapter 13 was done jointly with David Karger [74, 75] and Chapter 14 was the result of a collaboration with Alexander Hall and Christos Papadimitriou [63].

CHAPTER 15

Appendix: Quasi-concavity background

15.1 Concave and Quasi-concave functions

In this section, we define quasi-concave functions and extreme points, and state the problem of quasi-concave minimization and its hardness. Let C be a convex set.

Definition 15.1.1. *A function $f : C \to (-\infty, \infty)$ is* concave *if for all $x, y \in C$ and $\alpha \in [0,1]$,*

$$f(\alpha x + (1-\alpha)y) \geq \alpha f(x) + (1-\alpha) f(y).$$

A function $f : C \to (-\infty, \infty)$ is quasi-concave *if all of its upper level sets $L_\gamma = \{x \mid x \in C,\ f(x) \geq \gamma\}$ are convex.*

Remark: changing the sign of inequalities above yields the corresponding definition of *convex* and *quasi-convex* functions.

Definition 15.1.2. *A point x is an* extreme *point of the set C if it cannot be represented as a convex combination of two other points in C, namely $x = \alpha y + (1-\alpha)z$ for $y, z \in C$, $\alpha \in (0,1)$ $\Rightarrow$ $x = y = z$.*

The problem of (quasi-)concave minimization asks for the minimum of a (quasi-)concave function over a *feasible*, typically convex, set.

The following theorem about quasi-concave minimization over a convex set, stated without proof in Horst *et al.* [67], seems to be attributed to folklore. We include our proof—which is similar to the corresponding proof for concave functions in Bertsekas *et al.* [11], for completeness.

Theorem 15.1.3. *[67, 11] Let $C \subset \mathbb{R}^n$ be a compact convex set. A quasi-convex function $f : C \to \mathbb{R}$ that attains a maximum over C, attains the maximum at some extreme point of C.*

Proof. We use induction on the dimension m of the convex set. Suppose the theorem is true for sets of dimension at most $m - 1$. Then, in the case of dimension m the maximizer x^* is either in the relative interior of the set C or it is not. If it is, Lemma 15.1.4 establishes the result. Otherwise, x^* is in the relative boundary of C, which is of dimension one less than C and the result follows from the inductive hypothesis, noting that extreme points of the boundary are also extreme points of the original set. □

Clearly, the theorem also holds for *quasi-concave minimization*.

Lemma 15.1.4. *Let $C \subset \mathbb{R}^m$ be a compact convex set. A quasi-convex function $f : C \to \mathbb{R}$ that attains a maximum in the relative interior of C, attains the maximum at an extreme point of C.*

Proof. Suppose f attains a maximum at x^* in the relative interior [1] of C. Since C is compact, by Caratheodory/Krein-Milman's theorem [11] x^* is a convex combination of finitely many extreme points z_i of C. If $f(z_i) = f(x^*)$ for some z_i, then f attains a maximum at the extreme point z_i and we are done.

Otherwise, $f(z_i) < f(x^*)$ for all i so $\max_i f(z_i) = \beta < f(x^*)$. Consider the lower level set $L_\beta = \{x \mid f(x) \leq \beta\}$. It contains the extreme points z_i and is convex, therefore it contains the convex hull of these extreme points, $conv(\{z_i\})$. On the other hand, x^* is a convex combination of the z_i's, therefore $x^* \in conv(\{z_i\}) \subset L_\beta$. So $f(x^*) \leq \beta$, contradiction. Therefore, $f(z_i) = f(x^*)$ for at least one extreme point z_i, QED. □

Concave minimization is NP-hard even when restricted to concave quadratic functions over the hypercube [105]. In Section 5.2, we show that it is $(\log n)$-hard to approximate.

■ 15.2 Quasi-convexity of the probability-tail objective

We show that on its positive range the objective function $f : \mathbb{R}^m \to \mathbb{R}$, $f(\mathbf{x}) = \frac{t - \boldsymbol{\mu}^T \mathbf{x}}{\sqrt{\boldsymbol{\tau}^T \mathbf{x}}}$ is quasi-convex,[2] whence its maximum is attained at an extreme point of the feasible set [11], and further, it is attained at an extreme point of the projection of the feasible set onto the plane spanned by $(\boldsymbol{\mu}, \boldsymbol{\tau})$. This was observed in the context of stochastic shortest paths [99]; we provide a more formal proof here.

Lemma 15.2.1. *The function $f(\mathbf{x}) = \frac{t - \boldsymbol{\mu}^T \mathbf{x}}{\sqrt{\boldsymbol{\tau}^T \mathbf{x}}}$ is quasi-convex on its positive range.*

Proof. From the definition of quasi-convexity, we have to show that for all $\mathbf{x}, \mathbf{y} \in L_\lambda$ and $\alpha \in [0, 1]$, $\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in L_\lambda$, when $\lambda > 0$. To show this, we need to verify that

$$\frac{t - \boldsymbol{\mu}^T[\alpha\mathbf{x} + (1-\alpha)\mathbf{y}]}{\sqrt{\boldsymbol{\tau}^T[\alpha\mathbf{x} + (1-\alpha)\mathbf{y}]}} \leq \lambda$$

$$\Leftrightarrow \quad (t - \alpha\boldsymbol{\mu}^T\mathbf{x} - (1-\alpha)\boldsymbol{\mu}^T\mathbf{y})^2 \leq \alpha\lambda^2\boldsymbol{\tau}^T\mathbf{x} + (1-\alpha)\lambda^2\boldsymbol{\tau}^T\mathbf{y}.$$

Since $\alpha \in [0, 1]$, we have $\alpha(1-\alpha) \geq 0$, hence

$$-\alpha(1-\alpha)u^2 + 2\alpha(1-\alpha)uv - \alpha(1-\alpha)v^2 \leq 0 \qquad \forall u, v \in \mathbb{R}$$
$$\Rightarrow \quad \alpha^2u^2 + 2\alpha(1-\alpha)uv + (1-\alpha)^2v^2 \leq \alpha u^2 + (1-\alpha)v^2 \qquad \forall u, v.$$

[1] A point x is in the interior of a set C if there is a small ball with center x, which is contained in C. A point x is in the relative interior of a set C if there is a ball with center x of the same effective dimension as C, which is contained in C. For example, a line in a higher-dimensional space has effective dimension 1; thus it has an empty interior but a non-empty relative interior. The points in the closure of the set C which are not in its relative interior, form its relative boundary.

[2] A function g from a convex set C to $\mathbb{R}$ is *quasi-convex* if all its lower level sets $L_\lambda = \{\mathbf{x} \mid g(\mathbf{x}) \leq \lambda\}$ are convex.

Applying the above inequality with $u = \boldsymbol{\mu}^T\mathrm{x}$, $v = \boldsymbol{\mu}^T\mathrm{y}$, we get

$$\begin{aligned}
& (t - \alpha\boldsymbol{\mu}^T\mathrm{x} - (1-\alpha)\boldsymbol{\mu}^T\mathrm{y})^2 \\
= \; & t^2 + \alpha^2(\boldsymbol{\mu}^T\mathrm{x})^2 + 2\alpha(1-\alpha)(\boldsymbol{\mu}^T\mathrm{x})(\boldsymbol{\mu}^T\mathrm{y}) + (1-\alpha)^2(\boldsymbol{\mu}^T\mathrm{y})^2 - 2t\alpha\boldsymbol{\mu}^T\mathrm{x} - 2t(1-\alpha)\boldsymbol{\mu}^T\mathrm{y} \\
\leq \; & t^2 + \alpha(\boldsymbol{\mu}^T\mathrm{x})^2 + (1-\alpha)(\boldsymbol{\mu}^T y)^2 - 2t\alpha\boldsymbol{\mu}^T\mathrm{x} - 2t(1-\alpha)\boldsymbol{\mu}^T\mathrm{y} \\
= \; & \alpha(t - \boldsymbol{\mu}^T\mathrm{x})^2 + (1-\alpha)(t - \boldsymbol{\mu}^T\mathrm{y})^2 \\
\leq \; & \alpha\lambda^2\boldsymbol{\tau}^T\mathrm{x} + (1-\alpha)\lambda^2\boldsymbol{\tau}^T\mathrm{y},
\end{aligned}$$

where the last inequality follows from the fact that $\mathrm{x}, \mathrm{y} \in L_\lambda$. □

Corollary 15.2.2. *The maximum in the relaxed version of the nonconvex program (3.4) is an extreme point of the dominant*[3] *of the projection of polytope($\mathcal{F}$) onto $span(\boldsymbol{\mu}, \boldsymbol{\tau})$, provided there is a feasible point with positive objective value.*

Proof. As in Lemma 15.2.1, the projection $\bar{f} : \mathbb{R}^2 \to \mathbb{R}$ of the objective function f onto $span(\boldsymbol{\mu}, \boldsymbol{\tau})$, defined by $\bar{f}(\boldsymbol{\mu}^T\mathrm{x}, \boldsymbol{\tau}^T\mathrm{x}) = f(\mathrm{x})$, is quasi-convex. Denote by x_{max} the maximizer of f, then $\mathrm{z}_{max} = (\boldsymbol{\mu}^T\mathrm{x}_{max}, \boldsymbol{\tau}^T\mathrm{x}_{max})$ maximizes $\bar{f}$, hence z_{max} is an extreme point of the projection of polytope($\mathcal{F}$) onto $span(\boldsymbol{\mu}, \boldsymbol{\tau})$, by properties of projections [11]. In addition, $\bar{f}(\mathrm{z}) = \frac{t - z_1}{\sqrt{z_2}}$ is monotone decreasing in each coordinate on the function's positive range, which is where the maximum is achieved, hence z_{max} is an extreme point of the dominant of the feasible set projection. □

Since the extreme points of polytope($\mathcal{F}$) are in the feasible set $\mathcal{F}$, we get for free that the solution of the relaxed program also optimizes the integer version of the nonconvex program (3.4).

[3]The *dominant* of a set S is defined as the set of points that are coordinate-wise bigger than points in S, namely $\{\mathrm{y} \mid \mathrm{y} \geq \mathrm{x} \text{ for some } \mathrm{x} \in S\}$.

Bibliography

[1] D. Achlioptas, A. Clauset, D. Kempe, and C. Moore. On the bias of traceroute sampling; or, power-law degree distributions in regular graphs. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC'05)*, pages 694–703, 2005.

[2] H. Ackermann, A. Newman, H. Röglin, and B. Vöcking. Decision making based on approximate and smoothed pareto curves. In *Proc. of 16th ISAAC*, pages 675–684, 2005.

[3] H. Ackermann, A. Newman, H. Röglin, and B. Vöcking. Decision making based on approximate and smoothed pareto curves. In *Proceedings of 16th ISAAC*, pages 675–684, 2005.

[4] K. J. Arrow. The role of securities in the optimal allocation of risk-bearing. *Review of Economic Studies*, 31(2):91–96, 1964.

[5] R. Aumann. Agreeing to disagree. *Annals of Statistics*, 4:1236–1239, 1976.

[6] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[7] P. Barford, A. Bestavros, J. Byers, and M. Crovella. On the marginal utility of deploying measurement infrastructure. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[8] J. Berg, R. Forsythe, F. Nelson, and T. Rietz. Results from a Dozen Years of Election Futures Markets Research. *Handbook of Experimental Economics Results*, 2001.

[9] Y. Berstein, J. Lee, S. Onn, and R. Weismantel. Nonlinear optimization for matroid intersection and extensions. *Manuscript at arXiv:0807.3907*, 2008.

[10] D. Bertsekas. *Dynamic Programming and Optimal Control*, volume II, 2nd Edition. Athena Scientific, Belmont, MA, 2001.

[11] D. Bertsekas, A. Nedić, and A. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, Belmont, MA, 2003.

[12] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98:49–71, 2003.

[13] B. Bollobás and O. Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2004.

[14] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnády. The degree sequence of a scale-free random graph process. *Random Struct. Algorithms*, 18(3):279–290, 2001.

[15] T. Bonnesen and W. Fenchel. *Theory of Convex Bodies*. B C S Associates, Moscow, Idaho, 1988.

[16] J. Boyan and M. Mitzenmacher. Improved results for route planning in stochastic transportation networks. *Symposium of Discrete Algorithms*, 2001.

[17] D. Braess. ber ein paradoxon aus der verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968.

[18] J. Bruno, P. Downey, and G. N. Frederickson. Sequencing tasks with exponential service times to minimize the expected flow time or makespan. *J. ACM*, 28(1):100–113, 1981.

[19] P. Carstensen. *The complexity of some problems in parametric linear and combinatorial programming*. Ph.D. Dissertation, Mathematics Dept., Univ. of Michigan, Ann Arbor, Mich., 1983.

[20] Y. Chen, L. Fortnow, E. Nikolova, and D. M. Pennock. Betting on permutations. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 326–335, New York, NY, USA, 2007. ACM Press.

[21] F. Chung and L. Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1):91–113, 2003.

[22] V. Chvatal. Hard knapsack problems. *Operations Research*, 28:1402–1411, 1980.

[23] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 2:19–33, 1971.

[24] J. Conway. *A Course in Functional Analysis*. Springer-Verlag, 1990.

[25] C. Cooper, A. Frieze, K. Mehlhorn, and V. Priebe. Average-case complexity of shortest-paths problems in the vertex-potential model. *Random Struct. Algorithms*, 16(1):33–46, 2000.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[27] P. Cramton, Y. Shoham, and R. Steinberg. *Combinatorial Auctions*. The MIT Press, 2006.

[28] B. Dean. Personal communication. 2005.

[29] B. Dean, M. X. Goemans, and J. Vondrák. Approximating the stochastic knapsack: the benefit of adaptivity. In *Proceedings of the 45th Annual Symposium on Foundations of Computer Science*, pages 208–217, 2004.

[30] A. Deshpande and D. A. Spielman. Improved smoothed analysis of the shadow vertex simplex method. Preliminary version appeared in FOCS '05, 2005.

[31] G. Di Battista, T. Erlebach, A. Hall, M. Patrignani, M. Pizzonia, and T. Schank. Computing the types of the relationships between autonomous systems. *IEEE/ACM Transactions on Networking*, 15(2):267–280, April 2007.

[32] S. Dimitrov and R. Sami. Non-myopic strategies in prediction markets. In *EC '08: Proceedings of the 9th ACM conference on Electronic commerce*, pages 200–209, New York, NY, USA, 2008. ACM.

[33] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972.

[34] E. Elkind. True costs of cheap labor are hard to measure: Edge deletion and vcg payments in graphs. In *In Proceeding of 7th ACM conference on Electronic Commerce*, pages 108–116, 2005.

[35] T. Erlebach, A. Hall, A. Panconesi, and D. Vukadinovic. Cuts and disjoint paths in the valley-free path model of Internet BGP routing. In *CAAN*, pages 49–62, 2004.

[36] F. J. Fabozzi, P. N. Kolm, D. Pachamanova, and S. M. Focardi. *Robust Portfolio Optimization and Management.* John Wiley & Sons, Hoboken, New Jersey, 2007.

[37] Y. Fan, R. Kalaba, and I. J. E. Moore. Arriving on time. *Journal of Optimization Theory and Applications*, 127(3):497–513, 2005.

[38] U. Feige, V. S. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 461–471, Washington, DC, USA, 2007. IEEE Computer Society.

[39] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *In Proceedings of the 2002 ACM Symposium on Principles of Distributed Computing*, pages 173–182. ACM Press, 2002.

[40] J. Feigenbaum, V. Ramachandran, and M. Schapira. Incentive-compatible interdomain routing. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 130–139, New York, NY, USA, 2006. ACM.

[41] J. Feigenbaum, R. Sami, and S. Shenker. Mechanism design for policy routing. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 11–20, New York, NY, USA, 2004. ACM.

[42] A. D. Flaxman, A. Frieze, and M. Krivelevich. On the random 2-stage minimum spanning tree. *Random Structures & Algorithms*, 28(1):24–36, 2006.

[43] L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433, 1958.

[44] R. Forsythe, T. A. Rietz, and T. W. Ross. Wishes, expectations and actions: a survey on price formation in election stock markets. *Journal of Economic Behavior & Organization*, 39(1):83–110, May 1999.

[45] L. Fortnow, J. Kilian, D. M. Pennock, and M. P. Wellman. Betting boolean-style: a framework for trading in securities based on logical formulas. *Decis. Support Syst.*, 39(1):87–104, 2005.

[46] D. Friedman. Effective scoring rules for probabilistic forecasts. *Management Science*, 29(4):447–454, 1983.

[47] H. Fllmer and A. Schied. *Stochastic Finance: An Introduction in Discrete Time.* Walter de Gruyter, Berlin, 2004.

[48] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Trans. Networking*, 9(6):733–745, Dec 2001.

[49] M. Garey and D. Johnson. *Computers and Intractability.* Freeman Press, New York, 1979.

[50] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete problems. In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63, New York, NY, USA, 1974. ACM.

[51] L. E. Ghaoui, M. Oks, and F. Oustry. Worst-case value-at-risk and robust portfolio optimization: A conic programming approach. *Oper. Res.*, 51(4):543–556, 2003.

[52] A. Goel and P. Indyk. Stochastic load balancing and related problems. In *Proceedings of the 40th Symposium on Foundations of Computer Science*, 1999.

[53] A. Goel, K. Ramakrishnan, D. Kataria, and D. Logothetis. Efficient computation of delay-sensitive routes from one source to all destinations. In *Proceedings of IEEE Infocom*, 2001.

[54] G. Golub and C. V. Loan. *Matrix Computations.* Johns Hopkins University Press, 1996.

[55] R. Govindan and A. Reddy. An analysis of Internet inter-domain topology and route stability. In *Proc. IEEE INFOCOM*, pages 850–857, April 1997.

[56] V. Goyal. An FPTAS for minimizing a class of quasi-concave functions over a convex set. Technical Report Tepper WP 2008-E24, Carnegie Mellon University Tepper School of Business, 2008.

[57] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[58] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, Berlin Heidelberg, 1993.

[59] T. Groves. Incentives in Teams. *Econometrica*, 41:617–631, 1973.

[60] P. Gruber. *Convex and discrete geometry.* Springer, Berlin, 2007.

[61] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: approximation algorithms for stochastic optimization. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 365–372, Chicago, IL, USA, 2004.

[62] A. Gupta, M. Pál, R. Ravi, and A. Sinha. What about Wednesday? Approximation algorithms for multistage stochastic optimization. In *Proceedings of 8th APPROX*, pages 86–98, 2005.

[63] A. Hall, E. Nikolova, and C. Papadimitriou. Incentive-compatible interdomain routing with linear utilities. In *Lecture Notes in Computer Science (3rd International Workshop On Internet And Network Economics)*, volume 4858, pages 232–244, 2007.

[64] R. Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003.

[65] R. Hanson. Decision markets for policy advice. In E. Patashnik and A. Gerber, editors, *Promoting the General Welfare: American Democracy and the Political Economy of Government Performance.* Brookings Institution Press, Washington D.C., 2006. incollection George Decision Markets for Policy Advice.

[66] A. Hartemink. Personal communication. 2008.

[67] R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[68] N. Immorlica, D. Karger, M. Minkoff, and V. S. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 691–700, 2004.

[69] N. Immorlica, D. Karger, E. Nikolova, and R. Sami. First-price path auctions. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 203–212, New York, NY, USA, 2005. ACM Press.

[70] M. O. Jackson. A crash course in implementation theory. *Social Choice and Welfare*, 18:655–708, 1997.

[71] S. M. Kakade, A. T. Kalai, and K. Ligett. Playing games with approximation algorithms. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 546–555, New York, NY, USA, 2007. ACM.

[72] A. Kalai and S. Vempala. Efficient algorithms for on-line optimization. *Journal of Computer and System Sciences*, 71:291–307, 2005.

[73] D. Karger, R. Motwani, and G. D. S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18:82–98, 1997.

[74] D. Karger and E. Nikolova. Brief announcement: on the expected overpayment of VCG mechanisms in large networks. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 126–126, New York, NY, USA, 2005. ACM Press.

[75] D. Karger and E. Nikolova. On the expected VCG overpayment in large networks. In *45th IEEE Conference on Decision and Control*, pages 2831–2836, 2006.

[76] R. Karp. *Reducibility among combinatorial problems.* Plenum Press, 1972.

[77] I. Katriel, C. Kenyon-Mathieu, and E. Upfal. Commitment under uncertainty: Two-stage stochastic matching problems. *Theor. Comput. Sci.*, 408(2-3):213–223, 2008.

[78] J. A. Kelner and E. Nikolova. On the hardness and smoothed complexity of quasi-concave minimization. In *Proceedings of the 48st Annual Symposium on Foundations of Computer Science*, Providence, RI, USA, 2007.

[79] J. A. Kelner and D. A. Spielman. A randomized polynomial-time simplex algorithm for linear programming. *Electronic Colloquium on Computational Complexity*, Report No. 156, 2005.

[80] J. Kleinberg, Y. Rabani, and É. Tardos. Allocating bandwidth for bursty connections. *SIAM Journal on Computing*, 30(1):191–217, 2000.

[81] M. Krivelevich, Z. Nutov, and R. Yuster. Approximation algorithms for cycle packing problems. *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 556–561, 2005.

[82] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[83] L. Li, D. Alderson, W. Willinger, and J. Doyle. A first-principles approach to understanding the internet's router-level topology. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–14, New York, NY, USA, 2004. ACM.

[84] R. P. Loui. Optimal paths in graphs with stochastic or multidimentional weights. *Communications of the ACM*, 26:670–676, 1983.

[85] L. Lovász. Submodular functions and convexity. In *Mathematical Programming—The State of the Art, A. Bachem, M, Grőtschel and B. Korte, eds.*, pages 235–257. Springer-Verlag, 1983.

[86] B. Mangold, M. Dooley, G. W. Flake, H. Hoffman, T. Kasturi, D. M. Pennock, and R. Dornfest. The tech buzz game. *IEEE Computer*, 38(7):94–97, July 2005.

[87] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory.* Oxford University Press, New York, 1995.

[88] Mathematica Reference Guide. Gamma distribution. `http://mathworld.wolfram.com/GammaDistribution.html`.

[89] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4:414–424, 1979.

[90] M. Mehta. *Random Matrices*. Academic Press, Boston, MA, 1991.

[91] M. Mihail, C. Papadimitriou, and A. Saberi. On certain connectivity properties of the internet topology. *J. Comput. Syst. Sci.*, 72(2):239–251, 2006.

[92] P. Milgrom and C. Shannon. Monotone comparative statics. *Econometrica*, 62(1):157–180, 1994.

[93] E. D. Miller-Hooks and H. S. Mahmassani. Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science*, 34:198–215, 2000.

[94] P. Mirchandani and H. Soroush. Optimal paths in probabilistic networks: a case with temporary preferences. *Computers and Operations Research*, 12(4):365–381, 1985.

[95] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[96] R. Nau. Should scoring rules be "effective"? *Management Science*, 31(5):527–535, 1985.

[97] A. Nemirovski and A. Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2006.

[98] E. Nikolova, M. Brand, and D. R. Karger. Optimal route planning under uncertainty. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2006.

[99] E. Nikolova, J. A. Kelner, M. Brand, and M. Mitzenmacher. Stochastic shortest paths via quasi-convex maximization. In *Lecture Notes in Computer Science 4168 (ESA 2006)*, pages 552–563, Springer-Verlag, 2006.

[100] E. Nikolova and R. Sami. A strategic model for information markets. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 316–325, New York, NY, USA, 2007. ACM Press.

[101] N. Nisan. Bidding and allocation in combinatorial auctions. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 1–12, New York, NY, USA, 2000. ACM.

[102] S. Onn. Convex discrete optimization. *Encyclopedia of Optimization*, pages 513–550, 2009.

[103] C. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, 1991.

[104] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 86–92, Washington, DC, USA, 2000.

[105] P. M. Pardalos and G. Schnitger. Checking local optimality in constrained quadratic programming is NP-hard. *Operations Research Letters*, 7:33–35, 1988.

[106] D. Pennock. A dynamic parimutuel market for information aggregation. In *Proceedings of the Fourth Annual ACM Conference on Electronic Commerce (EC '04)*, June 2004.

[107] D. M. Pennock, S. Lawrence, C. L. Giles, and F. Å. Nielsen. The real power of artificial markets. *Science*, 291(5506):987–988, jan 2001.

[108] C. R. Plott and S. Sunder. Efficiency of experimental security markets with insider information: An application of rational-expectations models. *Journal of Political Economy*, 90(4):663–98, August 1982.

[109] C. R. Plott and S. Sunder. Rational expectations and the aggregation of diverse information in laboratory security markets. *Econometrica*, 56(5):1085–1118, September 1988.

[110] C. Polk, R. Hanson, J. Ledyard, and T. Ishikida. Policy analysis market: An electronic commerce application of a combinatorial information market. In *Proceedings of the Fourth Annual ACM Conference on Electronic Commerce (EC'03)*, pages 272–273, June 2003.

[111] G. Polychronopoulos and J. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27(2):133–143, 1996.

[112] M. Porembski. Cutting planes for low-rank-like concave minimization problems. *Operations Research*, 52(6):942–953, 2004.

[113] T. Radzik. Newton's method for fractional combinatorial optimization. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 659–669, 1992.

[114] R. Ravi and M. X. Goemans. The constrained minimum spanning tree problem (extended abstract). pages 66–75. Springer-Verlag, 1996.

[115] R. T. Rockafellar. Coherent approaches to risk in optimization under uncertainty. In *Tutorials in Operations Research INFORMS*, pages 38–61, 2007.

[116] Routeviews routing table. Snapshot from sep.15, 2003, 18:00 h. `http://www.routeviews.org/`, 2003.

[117] H. Safer, J. B. Orlin, and M. Dror. Fully polynomial approximation in multi-criteria combinatorial optimization. *MIT Working Paper*, February 2004.

[118] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artif. Intell.*, 135(1-2):1–54, 2002.

[119] D. B. Shmoys and C. Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *Journal of the ACM*, 53(6):978–1012, 2006.

[120] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos. Power laws and the as-level internet topology. *IEEE/ACM Trans. Netw.*, 11(4):514–524, 2003.

[121] A. Srinivasan. Approximation algorithms for stochastic and risk-averse optimization. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1305–1313, Philadelphia, PA, USA, 2007.

[122] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *Proceedings of the IEEE INFOCOM'02 Conference*, 2002.

[123] C. Swamy and D. B. Shmoys. Approximation algorithms for 2-stage stochastic optimization problems. *ACM SIGACT News*, 37(1):33–46, 2006.

[124] D. M. Topkis. *Supermodularity and complementarity.* Princeton University Press, Princeton, N.J., 1998.

[125] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[126] V. Vazirani. *Approximation Algorithms.* Springer, 2001.

[127] W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.

[128] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Oper. Res.*, 35(1):70–79, 1987.

[129] Wikipedia. `http://en.wikipedia.org/wiki/Autonomous_system_number`. *Autonomous system (Internet)*, 2009.

[130] N. Young, R. Tarjan, and J. Orlin. Faster parametric shortest path and minimum balance algorithms. *Networks*, 21(2):205–221, 1991.

[131] R. Yuster and Z. Nutov. Packing directed cycles efficiently. *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2004.